Aufbau eines Experten- und Informationssystems der Raumfahrt

Dipl.-Ing. Jürgen Leppich

Technische Universität Berlin
Fachbereich Verkehrswesen und angewandte Mechanik
Institut für Luft- und Raumfahrt

Jürgen Leppich
geboren am 24. März 1964 in Lünen

Aufbau eines Experten- und Informationssystems der Raumfahrt

Vom Fachbereich 10
- Verkehrswesen und Angewandte Mechanik -
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
genehmigte Dissertation

Wissenschaftliche Aussprache am 22.12.1999

Promotionsausschuss:

Vorsitzender:	**Prof. Horst Linde**
Berichter:	**Prof. Dr. phil. Roger Lo**
	Prof. Dr.-Ing. Erhard Konrad

Berlin 1999

D83

Danksagung

Die vorliegende Arbeit entstand im Fachgebiet Raumfahrzeugtechnik am Institut für Luft- und Raumfahrt der Technischen Universität Berlin.

Mein besonderer Dank gilt Herrn Prof. Dr. phil. Roger E. Lo (Technische Universität Berlin, Institut für Luft- und Raumfahrt, Fachgebiet Raumfahrzeugtechnik) für die Anregung zu dieser Arbeit und für die fortwährende Betreuung. Des weiteren danke ich Herrn Prof. Dr.-Ing. Erhard Konrad (Technische Universität Berlin, Institut für Angewandte Informatik, Fachgebiet Wissensbasierte Systeme) für die Bereitschaft, als zweiter Gutachter an diesem Promotionsverfahren mitzuwirken.

Besonders herzlich möchte ich meiner Frau Antje danken, die nicht nur bei der Korrektur der fertigen Arbeit selbst aktiv war, sondern auch bei der moralischen Betreuung während der Erstellung derselben unermüdlicher Antreiber und moralischer Aufrichter in einer Person war. Nicht zuletzt gilt mein Dank auch allen anderen Korrektoren, die sich bereit gefunden haben, den schwierigen Stoff komplett und intensiv zu lesen. Namentlich Herrn Marc Voslamber, Herrn Gerhard Kleintges und Herrn Markus Hentschel.

Ein besonderer Dank geht auch an Herrn Reinhard Nicolaus für die Unterstützung und tatkräftige Mithilfe bei allen großen und kleinen PROLOG Problemen. Ich möchte aber auch allen anderen Kollegen am Institut meinen Dank aussprechen, die meinen Aufenthalt dort zu einer besonders angenehmen Erfahrung in meinem Leben gemacht haben.

Jürgen Leppich, Berlin im Juli 1998

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen und Symbole

Abkürzungen	
1NF	First normal form (1.Normalform)
2NF	Second normal form (2.Normalform)
3NF	Third normal form (3. Normalform
4GL	Fourth generation language
4NF	Fourth normal form (4. Normalform)
AI	Artificial intelligence
AIM	Aerospace Information and Modeling
ANSI	American Nation Standards Institute
API	Application Program Interface
BCNF	Boyce-Codd normal form (Boyce-Codd Normalform)
CGI	Common Gateway Interface
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
DBI	Database Interface
DBMS	Database management system
DDL	Database Definition Language
DML	Data Manipulation Language
ERM	Entity Relationship Model
EVA	Extravehicular Activity
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IBM	International business machines
ISAPI	Internet Server programming interface
KI	Künstliche Intelligenz
LAN	Local Area Network
Mbps	Megabit per second
MVD	Multivariable Dependency
NSAPI	Netscape programming interface
PC	Personal computer
RDBMS	Relational database management system
SERM	Strukturiertes Entity Relationship Modell
SQL	Structured Query Language
URL	Universal Resource Locator
VDL	View Definition Language
WAN	Wide Area Network
WWW	World Wide Web

Symbole		
F	Schub	[N]
Isp	Spezifischer Impuls	[m/s]
N_K	Anzahl der Klassen	-
n_K	Anzahl der Attribute	-

Begriffserläuterung (Glossar)

Ablaufsteuerung	Die von der Inferenzmaschine verwendete Methode, mit der festgelegt wird, in welcher Reihenfolge die Schlussfolgerungen gezogen werden. Beispiele für Steuerungsmethoden sind Rückwärts- und Vorwärtsverkettung sowie Blackboard Aktionslisten /16/.
AppleTalk	Eine preisgünstige LAN Architektur, die in allen Apple Macintosh Computern und Laserdruckern bereits eingebaut ist. AppleTalk unterstützt Apple‘s LocalTalk Topologie (Kabelschema) ebenso wie Ethernet und IBM Token Ring. Mit Hilfe von AppleTalk kann man Macintosh Computer und Drucker miteinander verbinden und sogar PCs einbinden, wenn sie mit spezieller AppleTalk Hard- und Software ausgestattet sind /35/.
Attribut	Eine Eigenschaft eines Objekts. Zum Beispiel ist die Kreditwürdigkeit ein Attribut eines potentiellen Kreditnehmers. Attribute sind mit Werten in speziellen Fällen verknüpft. z.B. beträgt die Kreditwürdigkeit von H. Müller DM 120.000,- ./16/.
Backtracking	Suche in einem Baum nach dem Depth-First Prinzip /48/. Der Vorgang, eine Schlussfolgerungskette rückgängig zu machen, um einen anderen Suchpfad einzuschlagen. Planungsprobleme sind ein typisches Anwendungsgebiet für Backtracking-Strategien, die es ermöglichen, verschiedenen Lösungswege nacheinander auszuprobieren, bis sich ein Ergebnis als brauchbar herausstellt /16/. Rückkehr zu einem vorhergehenden Verzweigungspunkt um andere Alternativen zu untersuchen /4/.
Blackboard-Architektur	Ein Expertensystem-Entwurf, bei dem mehrere unabhängige Wissensbanken einen gemeinsamen Arbeitsspeicher abfragen, der als Blackboard bezeichnet wird. Ein auf einer Aktionsliste basierendes Steuerungssystem überprüft ständig alle möglichen unerledigten Aktionen und wählt die als nächste auszuführenden Aktion./16/.
Client	Die Client Seite einer Client-Server Architektur. Typischerweise ist ein Client eine Applikation (Programm) die auf einem PC oder einer Workstation läuft und davon abhängig ist, das spezielle Teilaufgaben auf einem Server ausgeführt werden. So ist z.B. ein E-Mail Client ein Programm, welches einen in die Lage versetzt, E-Mail zu empfangen und zu verschicken /35/.
Common Gateway Interface CGI	Eine Spezifikation für den Informationsaustausch zwischen einem WWW Server und einem CGI Programm. Ein CGI Programm ist jedes Programm, das in der Lage ist, Daten die der CGI Spezifikation entsprechen, zu empfangen und zurückzuliefern. Das Programm kann in jeder beliebigen Programmiersprache, einschließlich C, Perl, Java oder VisualBasic, erstellt werden. CGI Programme sind die am weitesten verbreite Möglichkeit für dynamische Interaktion zwischen Web Server (HTTP Server) und Benutzer. Zum Beispiel benutzen viele Web Seiten (HTML Dokumente), die Formulare enthalten, CGI Programme für die Auswertung der vom Benutzer gemachten Eingaben. Ein anderer, zunehmend weit verbreiteter, Weg um dynamische Rückmeldung für Web Benutzer bereitzustellen, ist die Integration von Skripten oder Programmen, die anstatt auf dem Web Server auf dem Computer des Benutzers laufen. Diese Programme können Java Applets, Java Skripte oder ActiveX Controls sein. Diese Techniken sind als clientseitige Lösungen bekannt, während die Benutzung von CGI eine serverseitige Lösung darstellt, weil die Abarbeitung auf dem Server stattfindet. Ein Problem bei der Benutzung von CGI ist, das jedes Mal wenn ein Skript gestartet wird, wird ein neuer Prozess generiert. Bei stark ausgelasteten Websites kann der Server dadurch merklich verlangsamt werden. Eine effektivere, aber auch schwerer zu implementierende Lösung, ist die Benutzung der Programmierschnittstelle (API) des Servers, wie beispielsweise ISAPI oder NSAPI. Eine weitere immer populärer werdende Lösung ist die Benutzung von Java Servlets /35/.
Constraint	Einschränkungen für eine Spalte in einer Tabelle einer relationalen Datenbank. Übliche Constraints sind NOT NULL = es muss ein Wert eingegeben werden, CHECK = der eingegebene Wert wird mit einem Ausdruck verglichen und nur akzeptiert wenn er diesen erfüllt, etc. /33/.

Daten data	(1) Verschiedene auf eine bestimmte Art formatierte Teile von Informationen. Jegliche Software wird in zwei grundsätzliche Kategorien eingeteilt: Programme und Daten. Programme sind eine Sammlung von Anweisungen zur Manipulation von Daten. Daten können in einer Vielzahl von Formen vorkommen – als Zahlen oder Text auf einem Stück Papier, als Bits und Bytes in einem elektronischen Speicher oder als Fakten im Kopf einer Person. Streng genommen sind Daten der Plural von Datum, einem einzelnen Stück Information. In der Praxis wird Daten jedoch für beide Formen, Singular und Plural, benutzt. (2) Der Begriff Daten wird auch oft benutzt um binäre maschinenlesbare Informationen von menschenlesbaren Textinformationen zu unterscheiden. Einige Programme unterscheiden z.B. zwischen Datendateien (data files = binäre Daten) und Textdateien (text files = ASCII Daten). (3) In Datenbankmanagementsystemen sind Datenfiles die Dateien, welche die eigentlichen Informationen enthalten, wohingegen Indexdateien und „data dictionaries“ administrative Informationen, auch Metadaten genannt, speichern /35/.
Datenbank Management System database management system DBMS	Eine Sammlung von Programmen, welche die Speicherung, Modifizierung und Extraktion von Informationen einer Datenbank ermöglichen. Es existieren viele verschiedene Arten von Datenbankmanagementsystemen, die von kleinen Systemen auf Personalcomputern bis hin zu riesigen Systemen auf Großrechnern reichen. Datenbankanwendungen umfassen unter anderem: - Computergestützte Bibliothekssysteme - Geldautomaten - Rechnergestützte Lagerhaltungssysteme - Flugreservierungssysteme Vom technischen Standpunkt aus, können sich DBMSs stark unterscheiden. Die Begriffe relational, Netzwerk, flach, hierarchisch beziehen sich alle auf die Art und Weise, wie ein DBMS seine Informationen intern organisiert. Der interne Aufbau bestimmt in entscheidendem Masse wie schnell und flexibel man auf die gespeicherten Informationen zugreifen kann. Anforderung von Informationen einer Datenbank werden in Form einer Anfrage (query) gestellt, bei der es sich um eine formalisierte Frage handelt. Die Anfrage: „SELECT ALL WHERE NAME = "SMITH" AND AGE > 35“ fordert zum Beispiel alle Datensätze (Rekords) in denen das Feld NAME gleich SMITH und das Feld AGE größer 35 ist, an. Die Regeln, nach denen eine Anfrage zusammengestellt wird, wird Abfragesprache (query language) genannt. Verschiedene DBMS unterstützen verschiedene Abfragesprachen, obwohl es einen Quasi-Standard namens SQL (structured query language) gibt. Hoch entwickelte Sprachen für die Verwaltung von Datenbanksystemen werden „fourth-generation languages“ oder abgekürzt 4GLs genannt. Die aus einer Datenbank abgerufenen Informationen können in einer Vielzahl von Formaten dargestellt werden. Die meisten DBMS beinhalten ein Berichterstellungsprogramm (report writer), welches die Informationen in Form eines Berichts abfragt. Viele enthalten auch eine Grafikkomponente mit der Informationen in Form von Grafiken und Diagrammen dargestellt werden können /35/.
Deduktion deduction	Logische Herleitung /48/.
Dialogkomponente	Modul, Teil eines Expertensystems, der die Schnittstelle zwischen Benutzer und Expertensystem darstellt. Langfristig wird bei Beratungs- bzw. Konsultationssystemen eine natürlichsprachige Interaktion angestrebt /46/.
Diskurswelt universe of discourse	Die Diskurswelt ist ein relevanter Ausschnitt der Realität. Die Abgrenzung dieses Ausschnitts erfolgt aus Sicht des jeweils durchzuführenden Modellierungsprojekts. Der nicht zur Diskurswelt gehörige Teil der Realität wird als Umwelt bezeichnet. Die Diskurswelt tritt mit der Umwelt über Umweltkontaktobjekte in Beziehung, die ebenfalls Gegenstand der Modellierung sind. Die Diskurswelt besteht aus dem Objektsystem und dem dafür vorgesehenen Zielsystem /11/.
Domäne domain	Ein Themenbereich oder Wissensgebiet. Sehr große Domänen sind z.B. die Medizin, die Technik und die Betriebswirtschaftslehre. Die derzeit vorhandenen Wissenssysteme liefern kompetente Ratschläge nur innerhalb sehr eng begrenzter Domänen /16/.

Datenbank database	Sammlung von zusammenhängenden, physisch gespeicherten Daten, die als eine Einheit betrachtet werden /18/. (1) Eine Sammlung von Informationen, welche so strukturiert wurde, das ein Computer schnell benötigte Datenteile auswählen kann. Man kann sich eine Datenbank als ein elektronisches Speichersystem vorstellen. Traditionelle Datenbanken sind in Felder (fields), Rekords (records) und Dateien (files) organisiert. Ein Feld ist ein einzelnes Stück Information; ein Rekord ist ein kompletter Satz von Feldern; eine Datei ist eine Sammlung von Rekords. Ein Telefonbuch ist ein Beispiel für eine Datei. Es enthält Listen von Rekords, welche aus drei Feldern bestehen: Name, Adresse und Telefonnummer. Ein alternatives Konzept einer Datenbank wird als Hypertext bezeichnet. In einer Hypertext Datenbank kann jedes Objekt, sei es eine Textpassage, ein Bild oder ein Film, mit einem anderen Objekt verbunden werden. Hypertext Datenbanken eignen sich besonders für die Speicherung großer Mengen grundverschiedener Informationen, aber sie sind nicht für numerische Auswertungen geeignet. Apple HyperCard ist ein solches Datenbankkonzept. Um auf die Informationen einer Datenbank zugreifen zu können, benötigt man ein Datenbankmanagementsystem (DBMS). Dabei handelt es sich um eine Sammlung von Programmen, die es einem ermöglicht Daten in die Datenbank einzugeben, zu organisieren und auszuwählen. (2) In zunehmendem Masse wird der Begriff Datenbank als Abkürzung für Datenbankmanagementsystem verwendet. Eine *Verteilte Datenbank (distributed database***)** ist eine Datenbank die aus zwei oder mehr Datenfiles besteht, die sich an verschiedenen Stellen im Netzwerk befinden. Weil die Datenbank verteilt ist, können verschiedene Benutzer auf die Daten zugreifen, ohne sich gegenseitig zu stören. Allerdings muss das DBMS die verstreuten Daten in regelmäßigen Abständen synchronisieren, um sicherzustellen, dass alle Benutzer konsistente Daten vorliegen haben. Eine *Flache Datenbank (flat-file database***)** ist ein relativ simples Datenbanksystem, in dem jede Datenbank in einer einzigen Tabelle enthalten ist. Im Kontrast dazu können relationale Datenbanksysteme mehrere Tabellen für die Informationsspeicherung benutzen, und jede Tabelle kann ein unterschiedliches Record Format haben. Relationale Systeme eigen sich besser für sehr große Anwendungen, allerdings sind flache Datenbanken bei kleineren Anwendungen ausreichend /35/.
Entity-Relationship-Modell ERM	Das dominierende Meta-Modell im Bereich der Anwendungsentwicklung betrieblicher Informationssysteme. Es wurde in seiner Grundform von P. P.-S Chen im Jahr 1976 vorgeschlagen und gehört zur Klasse der semantischen Datenmodelle. Mit zunehmendem Umfang der modellierten Schemata werden allerdings eine Reihe von Schwächen des ERM sichtbar. Eine der gravierendsten Schwächen ist die mangelnde Strukturierung der im ERM modellierten Schemata. Dies ist nicht nur ein Defizit des ERM als grafisches Darstellungsmittel sondern vor allem ein Defizit des ERM als Analyseinstrument /47/.
Ereignis event	Eine Aktion oder Vorkommnis, das von einem Programm wahrgenommen wird. Bei Ereignissen kann es sich um Benutzeraktionen, wie beispielsweise das Drücken einer „Maustaste“ oder einer Taste des „keyboard“ oder das Auftreten von Systemereignissen wie z.B. „running out of memory“ handeln. Die meisten modernen Anwendungsprogramme, speziell diejenigen die unter Apple Macintosh oder Windows Betriebssystemen laufen, werden als ereignisorientiert (event-driven) beschrieben, weil sie so entworfen wurden, das sie auf Ereignisse reagieren /35/.
Erklärungskomponente explanation component	Teil eines Expertensystems, das auf Anfrage erklärt, durch welche Regeln und Fakten Ergebnisse des Inferenzprozesses zustande gekommen sind, und warum bestimmte Aktionen durchgeführt wurden. Die Erklärungskomponente sollte mehrstufig sein, d.h. wenn der Benutzer an einer Stelle mehrmals die „Warum“-Frage stellt, muss sie z.B. die Schlussfolgerungskette aufrollen oder auch auf verschiedenen Ebenen beantwortet werden /46/.
Ethernet Ethertalk	Ein lokales Netzwerkprotokoll (LAN protocol) das 1976 von der XEROX Corporation zusammen mit DEC und Intel entwickelt wurde. Ethernet benutzt eine Bus oder Stern Topologie und unterstützt Datentransferraten von 10 Mbps. Die Ethernet Spezifikation diente als Basis für den IEEE Standard 802.3 welcher die physikalischen und unteren Softwareschichten (des OSI Modells) beschreibt. Ethernet benutzt die CSMA/CD Zugriffsmethode um simultane Zugriffe zu behandeln. Es ist einer der am häufigsten implementierten LAN Standards. Eine neuere 100Base-T oder Fast Ethernet genannte Version des Ethernet unterstützt Datentransferraten von 100 Mbps, und die neueste Version Gigabit Ethernet unterstützt Datenraten von 1 Gigabit (1000 Megabit) pro Sekunde /35/.

Expertensystem Expert System	Programm mit umfangreicher Wissensbasis über ein Spezialgebiet, das bequem genutzt werden kann /48/. Computerprogramm, das Problemstellungen mit einer einem Experten vergleichbaren Leistung lösen kann, insbesondere in Bereichen, wo das Wissen diffus ist und in denen langjährige Erfahrung zur Lösung von Aufgaben benötigt wird. Expertensysteme werden aber auch da eingesetzt, wo die algorithmische Lösung zu umfangreich wird, z.B. bei Schach. Ein Expertensystem sollte folgende Komponenten besitzen: Dialogkomponente, Problemlösungskomponente, Erklärungskomponente, Wissensakquisitionskomponente, Wissensbasis-, und Inferenzkomponente (entfällt bei PROLOG, da ein PROLOG Interpreter selber eine Inferenzmaschine ist) /46/.
Fremdschlüssel	Eine Spalte (oder eine Gruppe von Spalten) in einer Tabelle, die mit einem Primärschlüssel einer anderen Tabelle korrespondieren. Fremdschlüssel werden benutzt um Datenobjekte aus mehreren Tabellen zu kombinieren.
Heuristik Heuristic	Im allgemeinen benutzt für "rule of thumb" welche benutzt wird um intelligente Abschätzungen vorzunehmen, was zu tun ist. Bei Suchmethoden, mehr speziell, benutzt um die Kosten oder Entfernung eines Knotens zur Lösung abzuschätzen /4/. Regel/Prozedur zum Auffinden der Lösung eines Problems, die kein explizites Verhaltensmuster, sondern intuitive Verfahren vorgibt (im Gegensatz zum Algorithmus) /48/. Eine Faustregel oder andere Methode bzw. Vereinfachung, die die Suche in einem sehr großen Problemraum reduziert bzw. einschränkt. Zum Unterschied von Algorithmen garantieren Heuristiken nicht die richtige Lösung /16/.
Heuristiken	Vorgehensweisen bei Problemen, für deren Lösung keine eindeutigen Lösungsstrategien bekannt sind. In erster Linie „Daumenregeln“ auf der Grundlage subjektiver Erfahrungen und überlieferter Verhaltensweisen. Die Anwendung von Heuristiken ist vor allem in relativ unstrukturierten und schwer überschaubaren Problembereichen angebracht /46/.
Heuristische Suche heuristic search	Suchen unter Anwendung von Informationen über das Problem, die das Auffinden einer Lösung beschleunigen /48/. Suchmethoden, die Heuristiken benutzen /4/.
Horn Klausel	Aussagen in der logischen Programmierung, die durch „oder“ verknüpft sind, wobei höchstens eine Aussage positiv ist /16/.
HyperCard	Eine 1987 von Apple eingeführte Hypertext Programmierumgebung für den Macintosh. Das HyperCard Modell besteht aus Karten (cards) und Stapel (stacks) genannte Sammlungen von Karten. Man kann die Karten auf verschiedene Arten verbinden und wie einen Karteikasten durchblättern. Zusätzlich zu normalen Daten kann jede Karte Grafiken und Schaltknöpfe (buttons) enthalten, welche wiederum andere Ereignisse (Events) wie z.B. Töne und Filme auslösen. Jedes Objekt eines HyperCard Systems – Stapel (stack), Karte (card), Textfeld (field), Schaltknopf (button) oder Hintergrund (background) – kann mit einem Skript versehen werden. Ein Skript ist eine Sammlung von Instruktionen, die festlegen, welche Aktionen bei bestimmten Ereignissen (der Benutzer wählt ein Objekt aus oder drückt eine Schaltknopf) ausgeführt werden sollen. Das Schreiben von HyperCard Anwendungen wird als „authoring“ bezeichnet /35/.
Hypertext Markup Language HTML	Die Programmiersprache (authoring language) die benutzt wird um Dokumente für das World Wide Web zu erzeugen. HTML ähnelt SGML obwohl es keine Untermenge davon ist /35/.

Hypertext Transfer Protocol HTTP	Das dem WWW zugrundeliegende Protokoll. HTTP definiert wie Mitteilungen formatiert und übertragen werden und welche Aktionen Webserver und Browser als Reaktion auf verschiedene Kommandos ausführen sollen. Wenn man zum Beispiel eine URL im Browser eingibt, sendet dies ein HTTP Kommando an den Webserver, das ihn veranlasst, die entsprechende Webseite aufzufinden und zu übertragen. Der andere wichtige Standard, der kontrolliert wie das WWW arbeitet, ist HTML. HTML kontrolliert wie Webseiten formatiert und angezeigt werden. HTTP wird als „stateless“ Protokoll bezeichnet, weil jedes Kommando unabhängig, ohne Wissen über die Kommandos die vorher ausgeführt wurden, abgearbeitet wird. Das ist auch der Grund, warum es so schwierig ist Webseiten zu generieren, die intelligent auf Benutzereingaben reagieren können. Es wird versucht diesen Nachteil von HTTP mit einer Reihe von neuen Techniken (ActiveX, Java, Javaskript und Cookies) zu umgehen. Zur Zeit wird HTTP 1.1 von den meisten Webbrowsern und –servern unterstützt. Eine der wichtigsten Eigenschaften von HTTP 1.1 ist die Unterstützung von dauerhaften Verbindungen „persistent connections“. Das bedeutet, dass ein Browser mehrere Dateien über dieselbe Verbindung von einem Webserver empfangen kann, sobald er eine Verbindung zu einem Solchen hergestellt hat. Dies verbessert die Leistungsfähigkeit um 20% /35/.
Index	Physische Datenstruktur, die benutzt werden kann, um die Zugriffszeiten auf Tabellen zu verkürzen. Ein Index kann benutzt werden, um Eindeutigkeit innerhalb der Zeilen einer Tabelle sicherzustellen.
Inferenz	Schlussfolgerung, Schließen. Der Vorgang neue Fakten aus bereits bekannten Fakten abzuleiten. Eine Regel, die mit einer Inferenzregel (z.B. modus ponens) und einer bekannten Tatsache kombiniert wird, ergibt eine neue Tatsache /16/.
Inferenzregel Inference Rule	Regel die festlegt, was logischerweise aus existierenden Fakten geschlossen werden kann (z.B. modus ponens) /4/.
Inferenzmechanismus inference mechanism inference engine	Teil eines Expertensystems, der verantwortlich für das ziehen neuer Schlussfolgerungen aus aktuellen Daten und Zielen ist /4/. Gewinnt aus deklarativen und prozeduralem Wissen durch Anwendung von Inferenzregeln neues Wissen. Der Inferenzmechanismus ist die Komponente eines Expertensystems, das die Schlussfolgerungen im System nach verschiedenen Strategien durchführt. Es betrifft die Frage, wie die Interaktion zwischen den einzelnen Komponenten abläuft, wie Werte übergeben werden, welche Komponente wann arbeitet. Ein Interpreter der Computersprache PROLOG beinhaltet bereits einen mächtigen automatischen Inferenzmechanismus /46/.
Instanziierung	Die Spezifizierung bestimmter Werte. Eine bestimmte Person von bestimmten Geschlecht mit einer bestimmten Körpertemperatur ist eine Instanziierung des generischen Objekts Patient /16/. Die Bildung eines spezifischen Objektes als Ausprägung eines Objektes, das wiederum eine Klasse von Objekten mit bestimmten Eigenschaften beschreibt. In Prolog bezeichnet man ein Prädikat, dessen Variable Werte angenommen haben, als instanziiert. Ein instanziiertes Prädikat ist eine Ausprägung eines allgemein (mit Variablen) formulierten Prädikates. In der Wissensrepräsentation mit Frames bezeichnet man mit Instanziierung z.B. die Bildung des spezifischen Objektes "Wohnzimmer" aus dem Frame "Zimmer" /46/.
JOIN Bedingung	Informationen in einer relationalen Datenbank, die über mehrere Tabellen verteilt sind, werden mit JOIN Bedingungen in *einem* SQL Statement abgearbeitet. Die JOIN Bedingung sorgt dafür, dass Beziehungen zwischen Primär- und Fremdschlüsseln bei der Abfrage berücksichtigt werden.
Konfidenzfaktor	Subjektiver Sicherheitsfaktor. Eine zahlenmäßig ausgedrückte Einschätzung der Gewissheit eines Faktums oder einer Relation. Diese Zahlen verhalten sich anders als die Wahrscheinlichkeitswerte. Im allgemeinen wird beim Manipulieren von Konfidenzfaktoren methodisch weniger formal umgegangen als beim Kombinieren von Wahrscheinlichkeiten. Die meisten regelbasierten Systeme bevorzugen Konfidenzfaktoren gegenüber Wahrscheinlichkeiten /16/.

Künstliche Intelligenz KI	Ein Teilgebiet der Informatik, das sich mit den Konzepten und Methoden symbolischer Schlussfolgerungsprozesse durch einen Computer befasst sowie mit der symbolischen Darstellung des Wissens, das für die Schlussfolgerungen herangezogen wird. Dieses Gebiet untersucht die Möglichkeit, einen Computer zu einem Vorgehen zu veranlassen, das unter Menschen als intelligentes Vorgehen anerkannt wird /16/. Die Künstliche Intelligenz-Forschung befasst sich damit, Systeme zu entwickeln, die Verhalten zeigen, das man bei Menschen als intelligent bezeichnen würde (Turing-Test). Hierbei kann einmal die Erforschung und möglichst exakte Simulation menschlichen intelligenten Verhaltens betrieben werden, es können aber auch Systeme entwickelt werden, deren intelligentes Verhalten auf anderen Methoden und Hilfsmitteln als den von Menschen benutzten beruht /46/.
Mehrwertige Abhängigkeit multivalued dependency MVD	Eine mehrwertige Abhängigkeit beschreibt die Zuordnung einer Menge von Attributwerten zu einem anderen Attributwert. Sie besagt, dass innerhalb einer Relation r einem Attributwert von X einen Menge von Y-Werten zugeordnet wird, unabhängig von den Werten der restlichen Attribute von r /24/.
Modus Ponens	Eine Grundregel der Logik, die besagt: Wenn gilt: A impliziert B, und es gilt: A ist wahr, dann ist mit Sicherheit B ebenfalls wahr /16/.
Monotones Schließen	Ein Schlussfolgerungssystem, das auf der Annahme beruht, dass eine Tatsache, die als wahr erwiesen ist, sich im Verlauf des Schlussfolgerungsprozesses nicht mehr ändern kann /16/.
Nicht-monotones Schließen	Schlussfolgerungen, die revidiert werden können, wenn sich ein Wert im Verlauf einer Beratung ändert. Nicht-monotones Schließen lässt einen schnellen Wechsel von Werten eines Problems innerhalb kurzer Zeit zu. Wenn man z.B. ein On-Line-Expertensystem entwickeln will, das den Aktienmarkt beobachtet und über den Kauf von Aktien berät, würde man sich für ein nicht-monotones Schlussfolgerungsverfahren entscheiden, da es eine ständige Revision der Empfehlungen, entsprechend den Änderungen der Aktienpreise und des Angebots, zulässt /16/.
Normalisierung normalization	Der Prozess der Organisation von Daten beim Entwurf relationaler Datenbanken zur Vermeidung von Redundanzen. Normalisierung bedeutet in der Regel die Aufteilung von Datenbanken auf zwei oder mehr Tabellen und die Definition der Relationen zwischen den Tabellen. Das Ziel ist die Isolierung von Daten, so dass Ergänzungen, Löschungen und Modifikationen nur in einer Tabelle gemacht werden müssen und dann über die Relationen in der gesamten Datenbank weitergereicht werden. Es existieren drei wichtige Normalformen (jede mit steigendem Normalisierungsgrad). *1. Normalform (First Normal Form 1NF):* Jedes Feld einer Tabelle enthält verschiedene Informationen. In einer Liste von Angestellten würde jede Tabelle nur ein Geburtsdatum-Feld enthalten. *2. Normalform (Second Normal Form 2NF):* Kein Feldinhalt kann von einem anderen Feld hergeleitet werden. Wenn eine Tabelle bereits ein Geburtsdatum-Feld enthält, dann kann nicht noch ein Geburtsjahr-Feld hinzugefügt werden, da die Information dann redundant vorhanden wäre. *3. Normalform (Third Normal Form (3FN):* Es sind keine doppelten Informationen erlaubt. Wenn also zwei Tabellen ein Geburtsdatum-Feld benötigen, dann würde diese Information in einer eigenen Tabelle gespeichert werden und die beiden anderen Tabellen würden auf die Geburtsadatum-Information über ein Indexfeld in der Geburtsdatum-Tabelle zugreifen. Jede Änderung an einem Geburtsdatum würde automatisch in allen Tabellen wirksam, die eine Relation mit der Geburtsdatum-Tabelle besitzen. Es existieren noch weitere Normalisierungsstufen wie *die Boyce Codd Normalform (BCNF)*, die *4.Normalform (fourth normal form 4NF)* und die *5.Normalform (fifth normal form 5NF)*. Während Normalisierung auf der einen Seite die Verwaltung einer Datenbank vereinfacht, kann sie auf der anderen Seite die Datenbank sehr viel komplexer machen, da die Informationen auf sehr viele verschiedene Tabellen verteilt wird /35/.
O-A-W-Tripel	Objekt-Attribut-Wert-Tripel Eine Darstellungsmethode für Faktenwissen. Dies ist der allgemeinere und gebräuchlichere Begriff für die Relationen, die manchmal als Kontext-Parameter-Wert-Tripel bezeichnet werden. Ein Objekt ist eine tatsächliche begriffliche Einheit (Entity) in der Wissensdomäne des Benutzers. Attribute sind Eigenschaften, die mit Objekten assoziiert sind. Jedes Attribut kann verschiedene Werte annehmen /16/.

Objekt	Generell ist ein Objekt jeder Gegenstand, der individuell ausgewählt und manipuliert werden kann. Dies kann Formen und Bilder, die auf einem Monitor angezeigt werden, aber auch sehr viel weniger handfeste Softwareeinheiten einschließen. In der objektorientierten Programmierung ist ein Objekt beispielsweise eine gekapselte Einheit, die aus Daten und Prozeduren zur Manipulation des Objektes besteht /35/.
Objektsystem system of objects	Das Objektsystem umfasst die Objekte der Diskurswelt, die Umweltkontaktobjekte sowie die Beziehungen zwischen diesen Objekten. Innerhalb des Objektsystems werden anhand von Abgrenzungskriterien Teilsysteme (Informationssystem, Basissystem, Lenkungssystem, Leistungssystem, Anwendungssystem) abgegrenzt /11/.
Prädikatenlogik	Eine Erweiterung der Aussagenlogik. In der Prädikatenlogik wird jede elementare Einheit als Objekt bezeichnet. Aussagen über Objekte werden Prädikate genannt /16/.
Produktionsregel	Der in der kognitiven Psychologie verwendete Begriff zur Beschreibung einer Wenn-dann-Regel /16/. Regeln in der Form: wenn … Bedingung → dann … Ergebnis; bzw. Situation → Aktion /46/.
Produktionssystem production system	Ein menschliches oder Computersystem mit einer Datenbasis. Diese besteht aus Produktionsregeln und einem Steuerungsmechanismus, der die anwendbaren Produktionsregeln auswählt, während er einen Zielzustand anstrebt /16/. Ursprünglich von Post entwickelte, in der KI weiterentwickelte Form der Wissensrepräsentation, die auf der Idee der Produktionsregeln aufbaut. Das klassische Produktionssystem besteht aus Produktionsregeln. Eine Produktionsregel wird angewandt, wenn die beschriebene Situation auftritt. Die ebenfalls in der Produktionsregel beschriebene Aktion wird dann von Regelinterpreter ausgeführt. Viele Expertensysteme sind Produktionssysteme /46/.
PROLOG	In der KI verwendete Programmiersprache für logische Applikationen; beliebt zur Entwicklung von Expertensystemen /48/.
Rahmen frame	Art der Wissensrepräsentation, die deklarative und prozedurale Aspekte vereinigt. Zunächst von Minsky beschrieben für den Bereich der automatischen Bildverarbeitung und zurückgehend auf den Schemabegriff der Psychologie, später in weiteren Bereichen der Künstlichen Intelligenz eingesetzt. Ein Frame beschreibt ein Objekt zusammen mit all seinen zugehörigen Eigenschaften. Es enthält die wesentlichen Elemente einer Szene sowie ihre Beziehungen zueinander. Er besteht im wesentlichen aus seinem Namen und einer Reihe von Slots mit ihren Fillers. Frames bilden die Grundlage einer Reihe von Sprachen zur Wissensrepräsentation (KR-Sprachen), wie etwa FRL, KRL. Analoge Formen zu Frames sind Units, Skripten, Situation, Schemas, Prototypen, Property-Lists und Records /46/.
Rahmen Objekt Einheit frame	Wissensrepräsentationsschema, das ein Objekt mit einer Gruppe von Eigenschaften assoziiert (z.B. Fakten, Regeln, Default-Werte und Aktive Werte)(Jede Eigenschaft wird in einem Slot (Schlitz, Abteil) gespeichert. Ein Frame ist die Menge von Slots, die mit einem spezifischen Objekt in Relation stehen. Ein Frame entspricht einer Eigenschaftsliste bzw. einem Schema oder einem Record, wie diese Begriffe in der konventionellen Programmierung heißen /16/.
Reasoning	Urteilen, Beweisführung, logisches Schließen /48/.
Regel rule	Wissensdarstellung, die einen Zusammenhang, z.B. in der Form darstellt: wenn … Bedingung -> dann … Ergebnis; Ergebnis <- wenn … Bedingung erfüllt /46/.
Regelbasis rule base	Menge der zur Verfügung stehenden Operatoren, die auf die Daten anzuwenden sind /48/.
Relationales Datenbank Management System relational database management system RDBMS	Ein Datenbankmanagementsystem (DBMS), das Daten in Form von verknüpften Tabellen speichert. Relationale Datenbanken sind sehr mächtige Werkzeuge, da sie nur wenige Annahmen über die Beziehungen zwischen den Daten und wie die Daten aus der Datenbank abgefragt werden sollen, erfordern. Als Folge davon kann ein und dieselbe Datenbank auf verschiedenste Arten betrachtet werden. Ein wichtiges Merkmal relationaler Systeme ist, das eine Datenbank über mehrere Tabellen verteilt werden kann. Dies unterscheidet sie von flat-file Datenbanken, bei denen jede Datenbank in einer Tabelle gespeichert wird. Nahezu alle groß angelegten Datenbanksysteme sind RDBMS. Kleinere Datenbanksysteme hingegen benutzen andere Entwürfe die weniger Flexibilität bei den Abfragen gestatten /35/.

Resolution	Eine Methode, um herauszufinden, ob eine neue Tatsache anhand einer Anzahl vorgegebener logischer Aussagen gültig ist /48/. Die Inferenzstrategie, die in Logiksystemen angewandt wird, um die Gültigkeit einer Aussage zu bestimmen. Diese komplexe aber überaus effektive Methode erbringt den Beweis für die Gültigkeit einer Aussage, indem man feststellt, dass ein Widerspruch auftritt, wenn man die einzelnen Aussagen zu bestätigen sucht und eine davon eine Verneinung der zu beweisenden These ist /16/. Eine Inferenzregel und einfaches Überprüfungsverfahren basierend auf dieser Regel. Wird von PROLOG benutzt /46/.
Rückwärtsverkettung backward chaining goal driven backward reasoning	Inferenzmechanismus, bei dem vom Ziel ausgegangen wird und versucht wird, dieses durch Aufteilung in Subziele und Beweis der Subziele zu beweisen. Dieses Vorgehen wird solange rekursiv wiederholt, bis elementare Fakten erreicht sind, die entweder vorhanden oder nicht vorhanden sind /46/. Methode zur Problemlösung (Inferenz), welche von einem Ziel oder einer Hypothese ausgeht und, rückwärts vorgehend, Regeln benutzt um herauszufinden, welche Fakten notwendig sind, um das Ziel zu beweisen /4/. Argumentationsweise "von hinten nach vorn"; ausgehend von der Lösung wird versucht, den Ausgangszustand zu erreichen /48/.
Schlüssel	Logische Datenstruktur, welche über einen Index definiert wird, um Zeilen einer Tabelle eindeutig zu identifizieren.
Semantisches Netz	Repräsentationsformalismus, bestehend aus Knoten, die durch gerichtete, markierte Kanten verbunden sind. Knoten repräsentieren üblicherweise Konzepte. Die Kanten geben die Relationen zwischen diesen an. Über Anzahl und Art der Knoten- und Kantentypen herrscht weitgehend Uneinigkeit, es gibt allerdings einige allgemein verwendete Typen, wie "Ist -ein" (Is-a) und "Teil-von" (Part-of). Die Struktur des Semantischen Netzes soll die assoziativen Verbindungen zwischen Fakten darstellen, um direkten Zugriff auf "benachbartes" Wissen zu ermöglichen. Aufgrund dieser Eigenschaften werden semantische Netze auch als Assoziative Netze bezeichnet /46/.
Structured Query Language SQL	SQL ist eine standardisierte Abfragesprache für die Anforderung von Informationen von einer Datenbanken. Die Originalversion, genannt SEQUEL (Structured English Query Language) wurde 1974 und 1975 von einem IBM Forschungszentrum entworfen. SQL wurde zum ersten Mal 1979 von der Firma ORACLE in einem kommerziellen Datenbanksystem angeboten. Historisch bedingt war SQL die favorisierte Abfragesprache für Datenbankmanagementsysteme, die auf Minicomputern und Großrechnern liefen. In zunehmendem Masse wurde SQL jedoch auch von PC Datenbanksystemen unterstützt, weil es verteilte Datenbanken (Datenbanken, die über mehrere Computersysteme verteilt sind) erlaubt. Dies ermöglicht mehreren Benutzern in einem LAN dieselbe Datenbank simultan zu benutzen. Obwohl es verschiedene SQL Dialekte gibt, ist es nichtsdestotrotz das was heutzutage einer Standardabfragesprache am nächsten kommt. 1986 wurde eine rudimentäre Version von SQL von ANSI als der offizielle Standard bestätigt, aber die meisten SQL Versionen haben seitdem unzählige Erweiterungen zum ANSI Standard eingeführt. 1991 wurde der Standard vom ANSI nachgebessert. Dieser neue Standard wird als SAG SQL bezeichnet /35/.
Strukturiertes Entity-Relationship-Modell SERM	Erweiterung des ERM. Durch Erweiterung der Komplexität bei Relationen und der Einführung eines Existenzkriteriums wird eine tiefergehende Strukturierung der Daten erreicht. Der wesentliche Unterschied zum ERM besteht darin, dass Schlüsselreferenzen zwischen Datenobjekttypen primär unter dem Gesichtspunkt der darin enthaltenen Existenzabhängigkeiten analysiert werden. Dabei werden alle Paare von in Beziehung stehenden Datenobjekttypen nach dem Schema originär/abhängig geordnet. Bei der grafischen Darstellung im SER Diagramm wird der abhängige Datenobjekttyp rechts vom originären angeordnet. Dadurch entsteht ein quasi-hierarchischer Graph (gerichtet und azyklisch), der Existenzabhängigkeiten zwischen Datenobjekttypen sowie Folgen von Existenzabhängigkeiten klar visualisiert /12/.
Tabelle table	Die grundlegende Speichereinheit einer relationalen Datenbank. Daten die in Zeilen (rows) und Spalten (columns) angeordnet sind. Ein Tabellenblatt (Spreadsheet) ist zum Beispiel eine Tabelle. In Relationalen Datenbankmanagement Systemen sind alle Informationen in Form von Tabellen gespeichert /35/.

TCP/IP Transmission Control Protocol/Internet Protocol	Zusammenstellung von Kommunikationsprotokollen die benutzt wird, um Hosts über das Internet zu verbinden. TCP/IP benutzt mehrere Protokolle, von denen die wichtigsten TCP und IP sind. TCP/IP ist im UNIX System enthalten und ist das Protokoll das im Internet benutzt wird, was es zum de-facto-Standard für die Übertragung von Daten in Netzwerken macht. Selbst Netzwerkbetriebssysteme, die ihre eigenen Protokolle verwenden, wie zum Beispiel Netware, unterstützen TCP/IP /35/.
Unifizierung unification	Binden einer Variablen zur Laufzeit der Prozedur an einen Wert durch Pattern Matching /46/.
Vorwärtsverkettung forward chaining data driven forward reasoning	Schließt man von der Anfangssituation auf die Endsituation, so nennt man das Vorwärtsverkettung (Siehe auch Rückwärtsverkettung) /46/. Methode zur Problemlösung (Inferenz), welche Regeln auf Anfangsdaten anwendet und neue Schlussfolgerungen aus diesen Daten zieht /4/. Argumentationsweise "von vorn nach hinten"; ausgehend vom Startzustand wird versucht, eine Lösung zu finden/ /48/.
Wissensakquisition knowledge acquisition	Erweiterung der Wissensbasis eines Expertensystems durch direkte Interaktion zwischen Experten und Expertensystem, ohne Hilfe oder Zwischenschaltung des Knowledge-Engineers und ohne programmieren zu müssen. Wissenserwerb kann auch durch selbsttätiges Lernen des Expertensystems, durch Fallstudien und Protokolle, die das Expertensystem aufnimmt, geschehen /46/.
Wissensbasierte Systeme knowledge based systems	Computersysteme, die Wissen beinhalten und zwar neben Faktenwissen auch mit Unsicherheiten behaftetes, heuristisches und subjektives Wissen. Die Formulierung dieses Wissens ist die Aufgabe des Knowledge Engineering /46/.
Wissensbasis, Wissensbank knowledge base	Sammlung von Fakten und Regeln, die Spezialistenwissen in einem Expertensystem darstellen /4/. Die umfangreichste Komponente eines Expertensystems, die das für die Lösung von Problemstellungen in einem bestimmten Anwendungsgebiet benötigte Wissen enthält. Im Gegensatz dazu steht der Schlussfolgerungsmechanismus, dessen Aufgabe die Verarbeitung des in der Wissensbasis dargestellten Wissens ist /46/. Jener Teil eines Wissenssystems, der die Fakten und Heuristiken einer Domäne umfasst /16/.
Wissensdomäne knowledge domain	Ein abgegrenztes Wissensgebiet, wie z.B. die Konstruktion von Maschinen aus Modulen oder die Diagnose von Computersystemfehlern /46/.
Wissensingenieur knowledge engineer	Person, die die Wissensbasis des Expertensystems aufbaut, indem sie das Wissen von menschlichen Experten extrahiert und in eine für das Expertensystem verständliche Repräsentationsform umwandelt /46/.
Wissensrepräsentation knowledge representation	Systeminterne Darstellung des vorhandenen Wissens /48/. Die Methode, die zur Kodierung und Speicherung von Fakten und Relationen in einer Wissensbank angewandt wird. Semantische Netzwerke, O-A-W-Tripel, Produktionsregeln, Frames und logische Ausdrücke sind Modelle zur Wissensrepräsentation /16/. Das Forschungsgebiet der KI, das sich mit der Darstellung von Wissen im Computer beschäftigt /46/.
Wissenssystem, Wissensbasiertes System	Ein Computerprogramm, das Wissen und Inferenzverfahren zur Lösung schwieriger Probleme verwendet. Das Wissen, das notwendig ist um auf diesem Niveau zu operieren, sowie die verwendeten Inferenzverfahren können als Modell für das Expertentum erfahrener Fachleute angesehen werden. Im Gegensatz zu Expertensystemen werden wissensbasierte Systeme häufig zur Lösung kleiner schwieriger Probleme entworfen, während große komplexe Probleme echtes menschliches Expertentum erfordern. In vielen Fällen besteht der Nutzen kleiner Wissenssysteme vor allem in der Bedienerfreundlichkeit und weniger darin, dass sie Wissen abbilden, das durch ein konventionelles Programm schwer darzustellen wäre /16/.
World Wide Web WWW	Ein System von Internet Servern, welche speziell formatierte Dokumente unterstützen. Die Dokumente sind in einer HTML genannten Sprache formatiert, die Verbindungen mit anderen Dokumenten (links), genauso wie die Einbindung von Bildern, Audio und Videodateien unterstützt. Das heißt man kann von einem Dokument zum Anderen springen, indem man einfach auf sogenannte „hot spots“ klickt. Nicht alle Internet Server sind teil des WWW. Es existiert eine Vielzahl Browser genannter Programme, die den Zugriff auf das WWW erleichtern. Zwei der Populärsten sind der Netscape Navigator und Micosoft’s Internet Explorer /35/.

Anmerkungen

Tabellen- und Spaltennamen werden im Fixed-Font „Courier New“ dargestellt:

`Launcher`, `Stage_Propellant`

Auszüge aus Programmlistings werden mit dem Absatzformat „Sourcecode” dargestellt:

```
stage('Net Mass',Stage_ID,Mnet,'kg',['Net Mass',467,Y1,Y2,Y3,Y4,Y5,Y6],R):-
   not_used(467,R),
   assertz(used(467,R)),
   not_wanted(['Structure Mass','Recovery System Mass','Propulsion System Mass',
   'Usable Extra Propellant Mass','Guidance and Control Mass','Propellant Residuals Mass']),
   stage('Structure Mass',Stage_ID,M3,_,Y1,467),
   stage('Recovery System Mass',Stage_ID,M5,_,Y2,467),
   stage('Propulsion System Mass',Stage_ID,M4,_,Y3,467),
   stage('Usable Extra Propellant Mass',Stage_ID,M7,_,Y4,467),
   stage('Guidance and Control Mass',Stage_ID,M2,_,Y5,467),
   stage('Propellant Residuals Mass',Stage_ID,M6,_,Y6,467),
   Mnet is M2+M3+M4+M5+M6+M7.
```

Platzhalter sind in der Regel kursiv gedruckt (im folgenden Beispiel steht Objekttyp für einen der Objekttypen Launcher, Stage, Engine, etc.: *Objekttyp*_Geometry

Viele Fachbegriffe, sowohl aus dem Gebiet der Künstlichen Intelligenz, als auch der Raumfahrt sind englischsprachig. Es handelt sich hierbei um geläufige Ausdrücke, die sich teilweise einer Übersetzung entziehen und daher zum größten Teil nicht übersetzt sind. Wenn möglich wurde jedoch der entsprechenden deutschsprachige Begriff verwendet.

Abstrakt

In dieser Arbeit wird ein Informationssystem beschrieben, welches für Studium, Arbeit und Wissenschaft benötigte Daten aus der Raumfahrt zur Verfügung stellt. Die Informationen sind leicht zugänglich und werden in leicht zu handhabender Weise dargestellt. Zusätzlich zu dem reinen Faktenwissen (Datenbank) welches zu einem großen Teil aus Zahlenmaterial besteht, wurde Expertenwissen in das Informationssystem integriert. Dieses Expertenwissen dient zunächst dem Auffinden von Informationslücken, aber auch zur Auffüllung dieser Lücken, sowie zur Überprüfung des bereits gespeicherten Faktenwissens.

Die Speicherung des Faktenwissens ist als relationale Datenbank in Client/Server Architektur realisiert. Der Datenbankserver ist über das lokale Netzwerk oder per TCP/IP direkt erreichbar. Zusätzlich besteht die Möglichkeit durch ein CGI Interface über das World Wide Web (WWW) auf einen Teil der Datenobjekte zuzugreifen. Die Regelbasis besteht aus Formeln, Tabellen, Graphen und Ergebnissen von Simulationen. Es wurde aber auch die Vorgehensweise von menschlichen Raumfahrtexperten Lücken zu füllen nachgebildet. Hierzu wurden "rules of thumb" und Schätzgleichungen, die auf jahrelanger Erfahrung, bekanntem Wissen oder reiner Intuition beruhen, integriert.

Der Benutzer erhält auf seine Informationsanfrage einen Datensatz zu einem bestimmten Objekt der Datenbank (Träger, Stufe, Triebwerk etc.) mit allen Informationen aus dem Faktenwissen sowie aus den durch das Expertensystem erzeugten Daten. Zu beiden Informationen kann in einer Erklärungskomponente die Herkunft der Daten abgerufen werden. Beim Faktenwissen ist dies die Quelle der Daten (in der Regel ein Buch, Paper, Skript etc.), bei den durch das Expertensystem erzeugten Daten sind dies die angewandten Regeln.

Abstract

With this doctoral thesis an information system was created, which makes available aerospace data, considered to be necessary for studies, work, science and education. The information is easily accessible and delivered in a very comfortable way for the different user groups. In addition to the storage of pure "fact knowledge" (that forms the relational database) which consists mainly of numerical data, "expert knowledge" is integrated into the information system. This expert knowledge is used for detecting gaps in the "fact knowledge" and for filling these gaps by application of rules. Applying the "expert knowledge" to the stored facts validates the existing "fact knowledge".

The „fact knowledge“ is stored in a Relational Database which is accessed by a commercial Client/Server RDBMS. The Database Server may be accessed directly over LAN connections or via TCP/IP. In addition database connections via WWW are possible by using a CGI interface. The rule base consists of formulae, tables, graphs and results of simulations. The system simulates the problem solving techniques of human experts. Therefore rules of thumb and estimations relying on years of experience, existing knowledge or mere intuition were integrated.

As a result of a database query the user receives a set of data for a specific object (launcher, stage, engine etc.). This set includes all the information retrieved from the „fact knowledge“ as well as data created by the expert system. For both information types an explanatory component giving the origin of the data can be consulted. In the case of „facts“ that is the source (mainly a book, paper, etc.) in the case of data generated by the expert system this would mean the rules that were applied.

1 Einleitung und Übersicht

Diese Arbeit befasst sich mit dem Aufbau eines Experten- und Informationssystems am Institut für Luft- und Raumfahrt der Technischen Universität Berlin mit dem Ziel ein Datenbanksystem zu schaffen, das mit der Unterstützung von Techniken aus dem Bereich der Künstlichen Intelligenz die Möglichkeiten konventioneller Datenbanken erweitert. Es wird eine Verbindung zwischen Ingenieurwissenschaften und Informatik hergestellt, um aufzuzeigen, dass bei einer Nutzung von Techniken aus beiden Disziplinen ein signifikant besseres Ergebnis erzielt werden kann, als bei Beschränkung auf rein ingenieurwissenschaftliche Methoden. Obwohl auf den ersten Blick ein informationstechnisches Übergewicht vorliegt, sind an vielen Stellen ingenieurwissenschaftliche Überlegungen und Problemlösungstechniken eingeflossen und ein erfolgreicher Abschluss der Arbeit wäre ohne fundierte Ingenieurkenntnisse nicht möglich gewesen.

An dieser Stelle wird kurz auf die *Problemstellung* eingegangen, die mit dieser Arbeit behandelt wird. Es wird erläutert, wie die Handhabung verschiedener Informationen, die für die Studenten und Mitarbeiter des Fachgebiets für ihr Studium und ihre Arbeit notwendig sind, bisher gehandhabt wurde und wie der Zugriff auf diese Informationen bewerkstelligt wurde. Danach werden die möglichen Benutzer des Informations- und Expertensystems vorgestellt. Zum Schluss wird dann das "ideale" Informationssystem beschrieben, das von dieser Arbeit angestrebt, aber nicht erreicht wird. Daran anschließend wird die *Vorgehensweise* zur Lösung der Aufgabe vorgestellt. Die einzelnen Schritte bis zum fertigen Informations- und Expertensystem werden umrissen. Mögliche Probleme und ihre im Rahmen der Arbeit gefundenen Lösungen werden genannt und das Resultat der Arbeit wird übersichtlich und knapp vorgestellt.

1.1 Randbedingungen

Ausgangssituation

Am Institut für Luft- und Raumfahrt werden im Fachgebiet Raumfahrzeugtechnik für die Durchführung verschiedener Projekte Informationen aus dem Bereich der Raumfahrt benötigt. Am Institut liegen die Informationen in unterschiedlicher Form vor. Dies reicht von Büchern und Artikeln über elektronisch gespeicherte Informationen bis hin zu Expertenwissen, welches nur durch direkte Befragung des jeweiligen Experten erhalten werden kann. Neben den am Institut vorhandenen Wissensquellen, werden für die Recherche zu einem bestimmten Thema, innerhalb der Lehrveranstaltungen und wissenschaftlichen Arbeiten zusätzlich auch andere, teilweise externe, Informationsquellen genutzt. Die folgende Auflistung gibt einen kurzen Überblick über die verschiedenen Arten der Informationsquellen und Informationen:

- Skripte (in schriftlicher Form und als Computerfile) zu den Vorlesungen: Raketentechnik, Raumfahrtantriebe, Flugtreibstoffe, Raumfahrtplanung und -betrieb, außerdem zu den nicht mehr angebotenen Vorlesungen und nur in schriftlicher Form: Systemtechnik, Raumfahrtplanung, Raumfahrtbetrieb
- Fachgebietsbibliothek (Sammlung von Artikeln, Forschungsberichten, Vortragsabdrucke der großen Raumfahrtkongresse, Zeitungsausschnitten etc.)
- Institutsbibliothek / Universitätsbibliothek (Bücher, Mikrofiches, Zeitschriften)
- Öffentliche Bibliotheken (Bücher, Mikrofiches, Zeitschriften)
- Studien- und Diplomarbeiten in schriftlicher Form und als Computerfile
- Dissertationen in schriftlicher Form und als Computerfile
- Software (Simulationsprogramme, Rechenprogramme etc.)
- Internet Datenbanken und Recherchedienste (WWW, FTP etc.)
- Datenbanksoftware (HyperCard) mit Daten zu Triebwerken, Trägern, Kosten, etc.

Die vorhandene Hardware bestand aus einem Apple Macintosh Netz mit einigen nicht vernetzten PC-kompatiblen Computern.

Benutzer

Als Benutzer des zu erstellenden Informations- und Expertensystems kommen in erster Linie Studenten der Studienrichtung Raumfahrttechnik, sowie die am Institut beschäftigten wissenschaftlichen und sonstigen Mitarbeiter in Frage. Erst in zweiter Linie werden Mitarbeiter anderer Institute oder die Luft- und Raumfahrtindustrie als Nutzer berücksichtigt. Zuletzt sollte auch die breite Öffentlichkeit die Möglichkeit haben auf die Daten zurückzugreifen und das System zu benutzen.

Die Benutzung sollte ohne tiefere Kenntnisse der Software erfolgen können, jedoch wird der Wissensstand eines Studenten der Luft- und Raumfahrttechnik am Beginn des Hauptstudiums vorausgesetzt. Bei der Entwicklung des Expertensystems, insbesondere bei der Erklärungskomponente, werden gewisse Grundkenntnisse der Luft- und Raumfahrttechnik vorausgesetzt, um die Erläuterungen des Expertensystems nicht zu umfangreich werden zu lassen.

Auch für die Datensuche sollte der Benutzer bereits EDV-Kenntnisse mitbringen. Neben der als selbstverständlich vorausgesetzten Computer- und Softwarebedienung sollte der Benutzer Kenntnisse über den Aufbau von Datenbanken mitbringen und für tiefergehende Suchvorgänge die üblichen Abfragesprachen (wie zum Beispiel SQL) kennen.

Zielvorstellung

Das ideale Ergebnis dieser Arbeit wäre ein Informationssystem, welches den unbeschränkten Zugriff auf alle am Institut vorhandenen Informationen über ein einziges Benutzerinterface sicherstellt. Zusätzlich sollten alle nicht vorhandenen Informationen durch das Expertensystem generiert und die Antworten ausführlich erklärt und erläutert werden. Die Bedienung des Systems sollte Idealerweise mündlich über eine Spracherkennung oder wenn dies nicht möglich ist über ein natursprachliches zeichenorientiertes Interface erfolgen. Die Antworten des Systems sollten auf die gleiche Art wie die Eingabe erfolgen.

1.2 Vorgehensweise

Aufgabe dieser Arbeit ist es nun die bestehenden Daten und Informationen zu integrieren und durch Anwendung von KI-Methoden den einfachen Zugriff und die Konsistenz der Daten sicherzustellen. Ein weiterer Aspekt soll die Datenakquisition sein, wobei die Beschaffung der Daten auf Grund der großen Anzahl vorhandener Quellen kein Problem darstellt. Man erhält im Gegenteil sehr viele, zum Teil leider widersprüchliche Daten zu verschiedenen Aspekten der Luft- und Raumfahrttechnik in gedruckter Form sowie teilweise auch als Datenfiles (in der Regel Textdateien). Diese Daten müssen in die für die Speicherung notwendige Struktur gebracht werden. Ein weiteres Problem ist die Datenvalidierung. Auch hier kann man davon ausgehen, dass ein Einsatz von KI-Methoden eine Verbesserung der Leistungsfähigkeit bei der Datenvalidierung bringt.

In diesem Abschnitt wird die Vorgehensweise zur Erreichung des im vorhergehenden Abschnitt geschilderten Zieles beschrieben. Es wird der Zeitplan vorgestellt und die einzelnen Arbeiten, welche bis zur Fertigstellung durchgeführt wurden, aufgelistet. Die dabei aufgetretenen Probleme werden kurz genannt, um einen Eindruck von der Aufgabe zu bekommen. Die Detailprobleme werden in den jeweiligen Kapiteln ausführlich beschrieben. Dort werden auch die gefundenen Lösungen vorgestellt. Das fertige Resultat wird in Übersichtsform zusammengefasst. Auch hier werden die Details in den jeweiligen Abschnitten tiefergehend erläutert.

Der Aufbau der Datenbank sowie Konzeption und Programmierung des Expertensystems können getrennt betrachtet werden. Die Schnittstelle zwischen der Datenbank und dem Expertensystem besteht nur aus dem Austausch von Daten, welcher über SQL-Befehle abläuft. Da sich die Struktur der Datenbank im Laufe der Arbeit nur noch geringfügig geändert hat, waren auch die Anpassungen der Schnittstelle minimal.

Datenbank

Für den Datenbankbereich konzentrierten sich die Arbeiten auf die folgenden Aufgaben:

- Auswahl der Speicherform (objektorientierte oder relationale Datenbank)
- Auswahl der Informationen, die verfügbar sein sollen
- Klassifizierung und Anpassung an die Informationssystemstruktur
- Entwurf einer konsistenten Datenstruktur zu Speicherung der Informationen
- Übertragung der Datenstruktur auf eine Tabellenstruktur (Entity-Relationship Modell, Normalform)
- Installation und Einrichtung der Server und Client Software
- Anlegen der Datenbankobjekte auf dem Server (Tabellen, Spalten, Schlüssel, Indizes, etc.)
- Konvertierung/Umwandlung der vorhandenen Daten
- Auffinden und Verbessern von Fehlern
- Integration neuer Daten aus verschiedenen Quellen
- Anpassen der Struktur an neue Erfordernisse und Bedürfnisse

Der hauptsächliche Zeitaufwand konzentrierte sich dabei auf den Entwurf einer konsistenten Datenstruktur, welcher im Grunde bis heute nicht abgeschlossen ist. Jedes Mal, wenn neue Daten oder Objekte integriert wurden, musste eine sinnvolle Ergänzung der bestehenden Struktur, bis hin zu einem massiven Umbau einzelner Bereiche der Datenbank, erfolgen. Die Konvertierung und Eingabe vorhandener Daten nahm ebenfalls sehr viel Zeit in Anspruch, da eine Automatisierung aus mehreren Gründen - wie im folgenden Absatz geschildert - nicht immer möglich war.

Probleme Datenbank

Das Hauptproblem im Bereich der Datenbank liegt in der Komplexität der raumfahrttechnischen Objekte. Nahezu jede denkbare technische Lösung ist im Bereich der Raumfahrt vertreten. Während normalerweise alle Objekte eines bestimmten technischen Bereichs (z.B. Flugzeuge) einander relativ ähnlich sind, ist es im Bereich der Raumfahrt bereits innerhalb einer einzigen Objektklasse möglich, viele verschiedene technische Lösungen zu finden. Jeder noch so ausgefallene Entwurf wurde in der Regel auch verwirklicht und sollte möglichst mit allen Eigenheiten in der Datenbank gespeichert werden können. Außerdem sollten auch zukünftige Entwicklungen möglichst ohne größere Änderungen der Datenstruktur speicherbar sein. Aus diesen Gründen war es sehr schwierig eine geeignete Tabellenstruktur zu finden, die einigermaßen transparent und erweiterbar, aber auch unempfindlich gegenüber Bedienungsfehlern sein sollte. Hier wurde sehr viel Zeit in den Entwurf des Datenmodells und der Datenstruktur gesteckt, da sich die Unzulänglichkeiten einer gefundenen Struktur erst beim Auftauchen neuer zu speichernder Objekte und Daten herausstellte, welche plötzlich nicht mehr in das vermeintlich ideale Schema passten.

Bei der Übertragung der vorhandenen Daten bestand das Hauptproblem in der Identifikation eines bestimmten Objekts. Viele reale Objekte der Raumfahrt (Träger, Stufen, Triebwerke) liegen in mehreren nur leicht unterschiedlichen Versionen vor. Außerdem können mehrere unterschiedliche Bezeichnungen für ein und dasselbe Objekt gefunden werden. Auch widersprechen sich die Daten aus mehreren Quellen zu demselben Objekt teilweise erheblich. Dies liegt unter anderem an gezielter Desinformation, als auch an den zahlreichen Entwicklungsstufen eines Objekts. So findet man zum Beispiel bei Triebwerken häufig eine Mischung aus Daten der Vorentwürfe und Daten des tatsächlich gebauten Objekts. Da somit eine Zuordnung der neu einzutragenden Daten über den Namen des Objektes nur eingeschränkt möglich war, konnte dieser Vorgang auch nur schwer automatisiert werden.

Expertensystem

Das Expertensystem sollte in der Lage sein, die vorhandenen Daten zu überprüfen und auf Nachfrage bestimmte Attribute der Objekte zur Verfügung zu stellen. Außerdem sollten eventuelle Lücken im Faktenwissen erkannt und durch Anwendung von Regeln gefüllt werden. Der Benutzer soll zusätzlich die Information erhalten, woher die Daten stammen. Hierbei sind zwei Fälle möglich:

- Buch, Artikel oder sonstige schriftliche Quelle
- Anwendung von Regeln

Im ersten Fall erhält der Benutzer einen Literaturverweis, im Zweiten eine Erklärung zum Lösungsweg in Form der benutzten Ausgangsdaten und der Reihenfolge der angewandten Regeln, sowie Erläuterungen zu den Regeln selbst. Auf diese Art und Weise kann zu jedem in der Datenbank gespeicherten Zahlenwert die Quelle aus der dieser Wert stammt, angegeben werden.

Komponente zur Lückenerkennung

Damit die Regeln des Expertensystems angewendet werden können, um Lücken im Faktenwissen zu füllen, muss das Expertensystem wissen, was es nicht weiß. Dies hört sich trivial an, dieses Problem ist aber, bedingt durch die Struktur der Tabellen der AIM Datenbank, nicht ohne größeren Aufwand zu lösen. Auf Grund der universellen Speichermöglichkeiten ohne feste Spaltenstruktur für die zu speichernden Daten, kann anhand der Datenbankeinträge nicht direkt erkannt werden, welche Attribute oder Eigenschaften ein Objekt der Datenbank besitzen muss und welche nicht.

Um dennoch eine feste Zuordnung Objekt↔Eigenschaft vornehmen zu können, wurde zunächst versucht eine Klassenhierarchie, wie in der objektorientierten Programmierung, zu verwenden, um über die Zugehörigkeit eines Objektes zu einer bestimmten Klasse, durch die Vererbung von Attributen eine solche Bestimmung durchzuführen. Dies scheiterte jedoch an der mangelnden Strukturierbarkeit der Raumfahrtobjekte im Hinblick auf Eigenschaftsmengen, insbesondere bei den Antrieben. Ohne die Einführung von Standardwerten und Mehrfachvererbung war eine Klassenhierarchie nicht möglich. Aus diesen Gründen wurde auf eine Art Mustervergleich zurückgegriffen, bei dem Objekte mit gleichen Attributlisten zu Klassen zusammengefasst werden. Anhand der vorhandenen Attribute eines Objektes kann dann über Listenvergleich die Zugehörigkeit auf eine geringe Anzahl möglicher Klassen reduziert werden, und die diesen Klassen gemeinsamen Attribute sind dann diejenigen, die das untersuchte Objekt besitzen muss.

Komponente zur Lückenfüllung

Für die Komponente zur Ergänzung des Faktenwissens wurden Regeln aufgestellt, die basierend auf vorhandenen Attributwerten eines Objekts fehlende Attributwerte berechnen oder abschätzen können. Für die Formulierung der Regeln wurde auf verschiedene Informations- und Wissensquellen zurückgegriffen:

- Mathematische Funktionen (Formeln)
- "Rules of Thumb" und Expertenheuristiken (Regeln, Bedingungen, etc.)
- Tabellen, Grafiken, Zeichnungen, Bilder, Filme
- reine ASCII Texte und große formatierte Texte mit Objekten (Skripte)
- Programme und Simulationen (Rechenergebnisse, Vorgehensweisen, etc.)

Aus diesen Quellen wurden Regeln extrahiert und für eine Verwendung im Expertensystem formuliert. Dieses Expertenwissen kann nun benutzt werden, um das Faktenwissen zu ergänzen und zu überprüfen. So kann zum Beispiel eine Überprüfung neu einzugebender Daten schon während der Eingabe vorgenommen werden und der Benutzer auf eventuelle Unstimmigkeiten oder Eingabefehler hingewiesen werden. Es kann aber auch eine Überprüfung des bereits vorhandenen Faktenwissen vorgenommen werden, um mögliche Fehler zu berichtigen.

Probleme Expertensystem

Die Probleme beim Aufbau des Expertensystems waren vielfältig. Am Beginn der Expertensystementwicklung stand die Demonstration der grundsätzlichen Machbarkeit der Regelbasis in PROLOG. Das Funktionieren der Verbindung ORACLE-PROLOG war aufzuzeigen. Zusätzlich gab es noch einige prologspezifische Probleme. Die Vermeidung ungewollter Rekursion innerhalb der Regelbasis gestaltete sich aufwendiger als erwartet.

In PROLOG musste Wissen über die Datenbankstruktur integriert werden, damit auf das in den Tabellen der Datenbank gespeicherte Faktenwissen zugegriffen werden konnte. Die Form der Speicherung sollte möglichst tolerant gegenüber Änderungen in der Datenstruktur sein. Eine mögliche Umstrukturierung und Neueinteilung der Tabellen sollte keine Auswirkungen auf den PROLOG-Teil haben. Die Wissenserhebung (Extraktion von Regeln aus der Literatur, Ableiten von Wissen aus Simulationsergebnissen) konnte relativ einfach durchgeführt werden. Die Befragung der Experten des Instituts, die im Rahmen einer Diplomarbeit durchgeführt wurde, war jedoch schwieriger als erwartet (s.a. /55/).

Benutzerschnittstelle

Für den Zugriff auf die Informationen stehen nun eine Reihe von Standardwerkzeugen zur Verfügung, die auf verschiedenen Computer- und Betriebssystemplattformen verfügbar sind. Zusätzlich wurden eigene Programme entwickelt, die einen bequemen Zugriff erlauben. Die Programmierung wurde vom Autor, Mitarbeitern des Instituts und im Rahmen von Studien- und Diplomarbeiten durchgeführt. Die folgende Liste zeigt einen Ausschnitt der eingesetzten Software:

- Kommerzielle Software
  - ORACLE SQL*plus (Abfrage und Datenmanipulation)
  - ORACLE HyperCard Stack (Abfrage)
  - MS Access (Abfrage)
- HyperCard Stacks
  - AIM Datenbank (Abfrage und Datenmanipulation in tabellarischer Form)
  - Bibliothek Info (Abfrage)
  - Bibliothek (Abfrage und Datenmanipulation)
  - Benutzerfreundliche Datenmanipulationssoftware auf HyperCard Basis
- WWW-Schnittstelle (nur Abfragen)
  - Datenzugriff über Auswahllisten
  - Datenzugriff über ein 3D Schichtenmodell
  - Zugriff auf die Referenzen (Fachgebietsbibliothek)

Probleme Benutzerschnittstelle

Die für die Datenbank geltenden Probleme, ein Objekt der Raumfahrt nur über seinen Namen zu identifizieren, treffen natürlich auch für das Auffinden bestimmter Objekte zu. Da ein einfacher Stringvergleich der Namen nicht ausreicht, mussten neue Formen der Suche entwickelt werden. Naheliegend wäre es gewesen auf die Attribute und Attributwerte eines Objekts zuzugreifen und über einen paarweisen Vergleich gleicher Attribute eine Identifizierung durchzuführen. Dies scheiterte jedoch am Zeitrahmen.

Dagegen konnte die intuitive Bedienbarkeit, durch die im selben Zeitraum aufkommende starke Verbreitung des WWW, weitgehend realisiert werden. Die Integration einer Zugriffsmöglichkeit über das Internet mit Hilfe eines WWW-Browsers erleichtert den Zugriff auf die Daten ungemein. Die Benutzerschnittstelle ist dadurch betriebssystemunabhängig und kann weltweit genutzt werden.

1.3 Ergebnis

Das Resultat der Arbeit ist ein System von Einzelprogrammen, in dem die verschiedenen Informationsarten über leicht zu bedienende Oberflächen verfügbar sind. Die Programme verteilen sich auf folgende Module:

- Abfrage (Query)
  Auffinden von bestimmten Informationen, Unterstützung des Benutzers bei der Suche nach Informationen
- Darstellung (Display)
  Darstellung der Informationen, Darstellung der Ergebnisse einer Suche, Darstellung der verschiedenen Informationsarten, etc.
- Überprüfung (Validation)
  Datenvalidierung, Prüfung der Konsistenz und Richtigkeit der Daten
- Lernen (Acquisition)
  Daten- und Informationsbeschaffung, Daten- und Informationseingabe, Daten- und Informationskonvertierung, etc.

Aufgrund praktischer Überlegungen werden einige Bereiche von mehreren Programmen erfasst. Aus Zeitmangel konnte der Bereich „Lernen“ nicht realisiert werden. Die einzelnen Programme werden teilweise vom Expertensystem unterstützt, welches wiederum in kleinere spezialisierte Expertensysteme aufgeteilt werden kann. Der gesamte Expertensystemteil kann mit jeder beliebigen relationalen Datenbank eingesetzt werden. Allerdings sollte die Regelbasis natürlich zu dem Faktenwissen passen. Das heißt, es sollte sich um Regeln aus demselben Wissensgebiet handeln, welches auch in der relationalen Datenbank abgelegt ist. Weiterhin muss natürlich bei der Schnittstelle zwischen PROLOG und der Datenbank auf gleichlautende Objekt- und Attributbezeichnungen geachtet werden. Beides zusammen (Expertensystem und Datenbank) bildet ein wissensbasiertes System zur Informationsbereitstellung raumfahrtspezifischer Daten.

Die einzelnen Komponenten und der modulare Aufbau des Systems sind aus Abb. 1-1 ersichtlich. Dort sind auch die Schnittstellen zwischen den einzelnen Komponenten, deren Entwicklung einen Hauptteil der Arbeit ausmachte, zu erkennen. Durch die Verknüpfung von Standardsoftware (grau unterlegt) mit Eigenentwicklungen, zur Erweiterung der Funktionen und Integration der verschiedenen kommerziellen Softwareprodukte, war es möglich die Vorteile beider Arten zu nutzen. Standardisierte Zugriffswerkzeuge und weltweit einheitliche Schnittstellen mit individueller Gestaltung und Erweiterung der kommerziellen Software durch Verwendung von KI-Techniken.

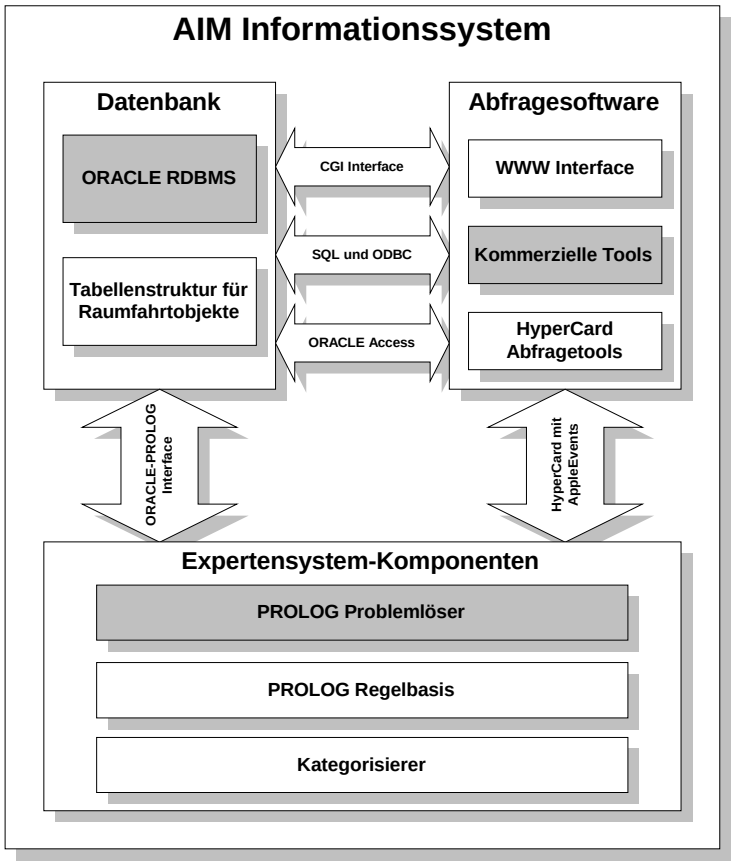


Abb. 1-1 Übersicht über das AIM Informationssystem (nicht am Institut entwickelte Komponenten sind grau unterlegt)

Datenbank

Das Faktenwissen ist als relationale Datenbank der Firma ORACLE in Client/Server Architektur realisiert. Der Datenbankserver (ohne Expertenwissen) ist über das lokale Ethertalk (Apple Macintosh) Netzwerk oder TCP/IP (mit der entsprechenden ORACLE Client Software) direkt erreichbar. Zusätzlich besteht die Möglichkeit durch ein CGI Interface über das World Wide Web (WWW) auf einen Teil der Datenobjekte zuzugreifen.

Die Datenbank enthält zur Zeit (Juni 1999) unter anderem 7573 Datenquellen (Bücher, Artikel, Studien- u. Diplomarbeiten, etc.) mit 650 Schlüsselwörtern, 7061 Personen (in ihrer Eigenschaft als Autoren, Astronauten/Kosmonauten), 873 Luft- und Raumfahrtorganisationen u. Firmen, 307 Trägersysteme mit 312 Stufen und 165 Nutzlastverkleidungen; 705 Triebwerke mit 124 Pumpen, 68 Turbinen und 107 Treibstoffkombinationen, 4075 Trägerstarts mit 4995 Missionen, 75 Satelliten und 76 Startplätze (inklusive Launch Pads), 13 Raumstationen mit 37 Stationsmodulen. Die Datenbank enthält viele Bilder und Zeichnungen mit und ohne Zuordnung. Zu den einzelnen Einträgen sind zum Teil ausführliche Textfiles verfügbar. Eine guten Überblick über die Anzahl der vorhandenen Daten gibt auch die Tabelle im Abschnitt 10.1 des Anhangs.

Expertensystem-Komponenten

Die Regelbasis besteht aus Formeln, Tabellen, Graphen und Ergebnissen von Simulationen, die in Regeln umgewandelt wurden. Es wurde aber auch versucht, die Vorgehensweise von menschlichen Experten nachzubilden. Hierzu wurden "rules of thumb" und Schätzgleichungen, die auf jahrelanger Erfahrung oder bekanntem Wissen beruhen, integriert. Die Regelbasis enthält zur Zeit etwa 400 Regeln zu Trägersystemen, Stufen und Triebwerken.

Die gesamte Regelbasis ist als PROLOG Programm (MacProlog32 V1.05o) erstellt worden. PROLOG dient auch als Inferenzmaschine/Problemlösungskomponente. Durch die Schnittstelle zwischen PROLOG und ORACLE kann das Programm direkt auf das Faktenwissen zurückgreifen. Aufgabe der Problemlösungskomponente ist das Auffüllen von Lücken im Faktenwissen durch Anwendung von Expertenregeln und die Überprüfung des vorhandenen Faktenwissen auf eventuelle Inkonsistenzen.

Der Benutzer erhält auf seine Informationsanfrage einen Datensatz zu einem bestimmten Objekt der Datenbank (Träger, Stufe, Triebwerk etc.) mit allen Informationen aus dem Faktenwissen sowie die durch das Expertensystem erzeugten Daten. Zu beiden Informationen kann in einer Erklärungskomponente die Herkunft der Daten abgerufen werden. Beim Faktenwissen ist dies die Quelle aus der das bestimmte Datum (Zahlenwert) stammt (in der Regel ein Buch, Paper, Skript etc.), bei durch die Problemlösungskomponente erzeugten Daten sind dies die angewandten Regeln.

Zusätzlich werden Daten, die einzelnen Regeln widersprechen oder deren Zahlenwert stark von den anderen Daten abweicht, als potentiell fehlerhaft gekennzeichnet. Der Benutzer erhält eine Liste mit den verdächtigen Einträgen der Datenbank, sowie die Regeln, gegen die verstoßen wurde.

2 Informationssysteme und Datenbanken

Im allgemeinen Teil, der die Abschnitte 2 und 3 umfasst, werden verschiedene Aspekte zu Informations- und Expertensystemen beleuchtet. Die in der Arbeit verwendete und im Bereich der Künstlichen Intelligenz bzw. bei Datenbanken übliche Nomenklatur wird eingeführt und erklärt. Der theoretische Unterbau für das aufgebaute Informations- und Expertensystem wird vorgestellt. In diesem Kapitel werden die verschiedenen Informationsarten identifiziert und beschrieben. Es werden die Informationen ausgewählt, die in dem zu erstellenden Informationssystem verarbeitet werden sollen. Außerdem wird auf die Lösungswege zur Speicherung und Verarbeitung der verschiedenen Informationsarten eingegangen.

Im kommerziellen Bereich findet man heutzutage eine Vielzahl an Datenbankprogrammen der verschiedensten Hersteller. Der Großteil sind Einzelplatzlösungen oder auch single-user Systeme. Einige wenige Hersteller bieten Client-Server Systeme mit "echten" Datenbankservern auf verschiedenen Betriebssystemen an. Von den auf dem Markt befindlichen Datenbanken zählt der überwiegende Teil zu den relationalen Datenbankmanagementsystemen. In den letzten Jahren sind verstärkt objektorientierte Datenbanken auf den Markt gekommen, die jedoch bisher in der Industrie nur wenig Anwendung finden. Dennoch soll hier auf beide Typen eingegangen werden. Die relationalen Datenbanken und die Datenmodellierung für relationale Datenbanken werden sehr ausführlich behandelt, da die AIM Datenbank ebenfalls auf einem relationalen Datenbanksystem basiert.

2.1 Objektorientierte Datenbanken

Objektorientierte Datenbanken werden seit Anfang der 90er Jahre im Bereich der EDV als Alternative oder Weiterentwicklung der bisherigen Datenspeicherung untersucht. Die Motivation für die Entwicklung gründet in der Suche nach leistungsfähiger Datenbankunterstützung im technisch-wissenschaftlichen Bereich. Insbesondere die komplexen und umfangreichen Datenstrukturen bei CAx Anwendungen (CAD, CAM, CAE, CIM) und bei geographischen Informationssystemen zeigen häufig die Unzulänglichkeiten der klassischen relationalen Datenbanksysteme auf /24/:

- Limitierte Datentypen
- Beschränkte Modellierbarkeit von Beziehungen
- Kein Zusammenhang von Code und Daten
- Limitierte Datenmanipulationsmöglichkeiten
- Schlechte Integration mit Programmiersprachen

In /17/ werden die Nachteile des relationalen Datenbankmodells RDM in Bezug auf die verschiedenen Problembereiche wie folgt aufgeführt:

- Datenmodellierung

  Attribute, die aus Wertemengen oder aus mehreren Komponenten bestehen, können simuliert, aber nicht direkt im RDM dargestellt werden.

  Beziehungen zwischen verschiedenen Relationen eines Objekttyps, "is-a"-Beziehungen zwischen verschiedenen Objekttypen und Objekt-Komponentenobjekt-Beziehungen können im relationalen Datenbankmodell nicht unterschieden werden. Sie werden jeweils über Fremdschlüssel-Beziehungen dargestellt.

- Datenbankentwurf

  Eingangsgrößen für den Datenbankentwurf sollten nicht nur Attribute und Abhängigkeiten zwischen Attributen sein, da mit diesen weder eine Anwendung vollständig beschrieben, noch eine verständliche und widerspruchsfreie Modellierung der Anwendung erreicht werden kann.

  Das Ergebnis der Dekomposition ist nicht abhängigkeitstreu, nicht minimal und sehr reihenfolgenabhängig. Teilweise werden Attribute völlig getrennter Objekttypen in ein Relationenschema aufgenommen, Informationen eines Objekttyps dagegen auf mehrere Relationenschemata verteilt oder gar nicht berücksichtigt.

  Zwar werden alle formalen Datenbankschema-Eigenschaften erreicht, trotzdem können Attribute unterschiedlicher Objekttypen nicht getrennt werden. Außerdem werden MVDs nicht berücksichtigt.

- Abfragesprachen

allgemein	Strukturmangel im Ergebnis
	Keine Unterstützung komplexer Strukturen in der Anfrageformulierung
	Notwendigkeit expliziter Verbundoperationen
SQL	Mangel an Orthogonalität
	Keine formale Definition
	Konzeptionsfehler
	Relationales Modell nicht voll unterstützt

- Update-Operationen

  Durch die Verteilung der Daten auf unter Umständen sehr viele Tabellen, werden umfangreiche Update-Operationen erschwert. Ein automatisches Einhalten der Konsistenzbedingungen wird von den meisten RDBMS nicht unterstützt.

- Optimierung

  Die Optimierungsmöglichkeiten bei relationalen Datenbanken sind sehr beschränkt. In der Regel hängt die Art der Optimierung vom verwendeten DBMS ab. Eine gezielte Optimierung auf geringen Speicherbedarf oder hohe Verarbeitungsgeschwindigkeit ist über die Datenmodellierung nur sehr eingeschränkt möglich

Objektorientierte Datenbanken vereinigen Einflüsse und Konzepte aus mehreren unterschiedlichen Bereichen der Informatik (Programmiersprachen, Softwaretechnik, Künstliche Intelligenz, Datenbanken bzw. Datenmodellierung). In Kurzform basiert Objekt-Orientierung auf den folgenden fünf Prinzipien /24/:

- Entitäten einer gegebenen Anwendung werden als Objekte mit eigener Identität modelliert, welche sowohl logisch als auch physisch von dem Wert des Objektes unterschieden wird (z.B. Zeiger auf das Objekt)
- Jedes Objekt kapselt Struktur und Verhalten. Die Struktur wird über Attribute beschrieben, deren Werte einfach, zusammengesetzt oder Referenzen sein können. Die Ausprägung aller Attribute eines Objektes zu einem Zeitpunkt ergibt den Zustand des Objektes, das Verhalten wird über Methoden beschrieben
- Auf den Zustand eines Objektes kann von außen durch das Verschicken einer *Message* an das Objekt zugegriffen werden (Methodenaufruf)
- Objekte mit gemeinsamer Struktur und gemeinsamem Verhalten werden in Klassen gruppiert, jedes Objekt ist Instanz einer Klasse
- Eine Klasse kann als Spezialisierung einer oder mehrerer anderer Klassen definiert werden. Unterklassen erben dabei Struktur und Verhalten ihrer Oberklassen

In der Regel werden objektorientierte Datenbanken (OODB) in enger Anlehnung an eine objektorientierte Programmiersprache implementiert (z.B. ObjectStore benutzt C++) Es wird unterscheiden zwischen dauerhaften (persistent) und veränderlichen (transient) Objekten. Die dauerhaften Objekte werden von der OODB verwaltet /18/.

Auf den ersten Blick spricht vieles für die Verwendung einer objektorientierten Datenbank im AIM System. Viele Eigenschaften der objektorientierten Programmierung finden sich in Methoden der KI wieder. Wie in Kap. 1 beschrieben werden OOP Methoden unter anderem in der Wissensrepräsentation benutzt, wobei das Schlüsselkonzept der OOP, die Vererbung, dort ebenfalls intensiv eingesetzt wird.

Zu Beginn der Arbeiten am AIM System wurde versucht, eine Objekthierarchie mit Vererbung als Datenstruktur für die in der Raumfahrt verwendeten Triebwerke zu entwerfen. Diese Struktur scheiterte im wesentlichen an der Vererbung, da die Objektausprägungen in der Raumfahrt und insbesondere bei den Triebwerken sehr vielfältig sind. Dies wird noch ausführlicher im Kap 5.1 erläutert. Quintessenz der Bemühungen war jedoch die Erkenntnis, dass Einteilungen der Triebwerke im Bereich der Raumfahrt auf verschiedenen Kriterien beruhen (z.B. Treibstoffe, Leistung, Bauweisen) und jeweils unterschiedliche Hierarchien ergeben. All diese Einteilungen werden von den Ingenieuren benutzt und es gibt keine allgemeingültige Einteilung, da alle Einteilungen je nach Bedarf und Betrachtungsweise sehr gut benutzt werden können. Für die Einteilung in Objekte und die Zuordnung von Attributen, sowie hauptsächlich für sinnvolle Vererbungsmechanismen, muss man sich für eine Methode der Objekteinteilung entscheiden. Tut man dies, so ist die daraus resultierende Hierarchie nicht mehr von allen Raumfahrtingenieuren nachvollziehbar. Außerdem tritt innerhalb der Hierarchie bei der Vererbung der Fall auf, das ein Objekt zwar im Hinblick auf die gewählte Einteilungsmethodik eine Spezialisierung eines mehr allgemeinen übergeordneten Objekts ist, und somit die Eigenschaften und Methoden erben könnte, unter Berücksichtigung einer anderen Einteilungsmethode, aber bestimmte Teilbereiche der Eigenschaften in der Vererbung überhaupt keinen Sinn machen. Dies ging bei dem Versuch der Einteilung soweit, dass man zu dem Schluss kommen musste, dass eine konsistente Klassenhierarchie ohne Standardwerte und Mehrfachvererbung im Bereich der Antriebe nicht machbar ist. Diese Erkenntnis lässt sich auf alle anderen Objekte der Raumfahrt übertragen.

Zusätzlich kann man sagen, dass eine objektorientierte Datenbank erst in Verbindung mit einer objektorientierten Programmiersprache ihre volle Leistungsfähigkeit ausspielen kann. Die Verwendung einer solchen Programmiersprache für die Softwareentwicklung innerhalb dieser Arbeit war nicht vorgesehen. Auch muss man sagen, dass nahezu alle verfügbaren OODBs nicht für Apple Macintosh Computer verfügbar sind und die Finanzierung einer solchen Datenbank für das Institut auf Grund der knappen Mittel nicht möglich war.

Unter Berücksichtigung der oben genannten Argumente wurde entschieden, keine objektorientierte Datenbank zu verwenden.

2.2 Relationale Datenbanken

Relationale Datenbanken sind Werkzeuge, um große Mengen an Daten in einer Form zu speichern, die es ermöglicht Informationen aus den Daten zu gewinnen. Eine relationale Datenbank besteht aus Tabellen. Eine Relation ist das mathematische Äquivalent einer Tabelle /18/. Als Alternativen zu relationalen Datenbanken bieten sich einzelne Dateien (flat files), programmspezifische Dateibehandlung, Hierarchien, Netzwerke und die bereits besprochenen objektorientierten Datenbanken, an. In /24/ werden die Vorteile von relationalen Datenbanken wie folgt zusammengefasst:

- Permanente Speichermechanismen
- Schnelle Abfragen
- Geräteunabhängige Formate
- Programmunabhängige Datenspeicherung
- Ausgereifte Fehler- und Transaktionsbehandlung

Relationale Datenbanken sind weit verbreitet, da größere Datenmengen in der Regel tabellarisch organisiert werden und Daten in tabellarischer Form vom Benutzer nur schwer überblickt und erfasst werden können. Als Hilfsprogramm für Manipulation und Organisation tabellarisch gespeicherter Daten werden daher Datenbanksysteme benötigt. Nicht geeignet sind relationale Datenbanken für sehr große Mengen unstrukturierter Rohdaten. Bilder, Filme und Töne (nahezu alles außer Text und Zahlen) können nur unter Inkaufnahme größerer Schwierigkeiten in relationalen Datenbanken gespeichert werden. Komplexe Berechnungen und sich dynamisch verändernde Strukturen stellen relationale Datenbanken ebenfalls vor größere Probleme.

RDBMS

Für den Zugriff auf die Informationen und die Manipulation der Daten einer relationalen Datenbank ist ein RDBMS (Relational Database Management System) verantwortlich. Ein RDBMS ist eine Sammlung von Programmen die den Benutzer in die Lage versetzen eine Datenbank anzulegen und zu warten. Die hauptsächlichen Anforderungen an RDBMS werden in /18/ wie folgt zusammengefasst:

- Datenabstraktion
  Der Benutzer muss sich nur mit der konzeptuellen Darstellung der Daten befassen, die stark von der eigentlichen Speicherung der Daten abweichen kann
- Grundlegende Datenbankarchitektur
  Schutz des Benutzers gegenüber low-level Details durch verschiedene Softwareschichten. Meta-Daten (z.B. Informationen über die Tabellenstruktur, Datenkatalog) sind verfügbar
- Mehrbenutzerunterstützung
  Datenaustausch, gleichzeitiger Zugriff, Transaktionsverarbeitung, mehrfache Datenansichten
- Unterstützung verschiedener Benutzertypen
  Administratoren (DBA), Datenbankdesigner, Endbenutzer, gelegentliche Endbenutzer, unerfahrene Endbenutzer, etc.
- Verschiedene Zugriffsmöglichkeiten auf das System
  Abfragesprachen, Programmiersprachen, Abfragemasken, Menusteuerung
- Kontrolle von Informationsredundanzen
- Datenbanksicherheit (Verhinderung nicht autorisierter Zugriffe)
- Erzwingung der Einhaltung von Integritätsbeschränkungen
- Datensicherung und -wiederherstellung

Codd Regeln

Einen sehr viel genaueren Einblick in die Eigenschaften eines "wahren" relationalen DBMS geben die zwölf Regeln, die der Erfinder des relationalen Modells Dr. E.F. Codd 1985 in einem zweiteiligen Artikel in der ComputerWorld /7/, /8/ zu Protokoll gegeben hat. An diesen Regeln kann man ablesen, ob ein DBMS das begehrte R auch wirklich verdient. Sie basieren sämtlich auf der Basisregel Null: Ein RDBMS muss fähig sein, Datenbanken vollständig durch seine relationalen Fähigkeiten zu verwalten. Das bedeutet, dass es nicht ausreicht, wenn beispielsweise an ein hierarchisches System gewisse relationale Funktionen quasi angebaut werden, man aber immer noch auf nicht-relationale Elemente angewiesen ist, um die Produktion durchzuführen. Diese Regeln stellen eine sehr nützliche Messlatte für die Bewertung eines relationalen Systems dar. Codd erwähnt allerdings auch, dass bis heute kein vollständig den Regeln entsprechendes voll relationales System existiert. Insbesondere die Regeln 6, 9, 10, 11 und 12 sind sehr schwer zu erfüllen.

Im Folgenden ist eine in der Computerwoche /37/ abgedruckte, und mit Anmerkungen versehene, Übersetzung der Codd Regeln wiedergegeben. Das Original kann in verschiedenen Quellen /10/, /36/ nachgelesen werden.

1. Information

Alle Informationen in einer relationalen Datenbasis werden auf dem logischen Level explizit und ausschließlich als Werte in Tabellen repräsentiert. Dies gilt nicht bloß für die Daten, die den Benutzern zur Verfügung stehen, sondern auch für Systeminformationen, die im Katalog gespeichert werden. So wird eine wirklich umfassend einheitliche Darstellung für alle erreicht, die mit dem DBMS arbeiten müssen: Benutzer, Administratoren und Programmierer.

2. Garantierter Zugriff

Auf jedes einzelne Datenelement (Tabelleneintrag) in einer relationalen Datenbasis kann mittels einer Kombination von Tabellenname, Primärschlüsselwert und Spaltenname logisch zugegriffen werden. Wichtig ist hier, dass der Zugriff völlig unabhängig von einer rechnerorientierten Adressierung und damit von der physischen Speicherung ist.

3. Systematische Behandlung von NULL Werten

Um fehlende Informationen in einer systematischen Art darzustellen, und zwar unabhängig vom Datentyp, muss ein RDBMS NULL-Werte unterstützen. NULL ist nicht mit 0 oder einer leeren Zeichenkette identisch, sondern bedeutet, dass für den betreffenden Tabellenwert kein sinnvoller Eintrag vorliegt - er ist (noch) undefiniert. Es muss natürlich möglich sein, bei der Tabellendefinition NULL für bestimmte Spalten zu verbieten. Dies ist zum Beispiel bei einem Primärschlüssel der Fall.

Heutzutage, das heißt in seiner Version 2 des relationalen Modells, fordert Codd zwei mögliche Werte für NULL: Einer gibt an, dass der betreffende Eintrag unbekannt ist, aber im Prinzip existieren kann, und der andere, dass ein Eintrag zu diesem Primärschlüsselwert nicht zugelassen ist, etwa bei der Spalte Heiratsdatum für Ledige.

4. Dynamischer Online Katalog basierend auf dem relationalen Modell

Die Beschreibung der Datenbanken auf dem logischen Level wird wie gewöhnliche Daten repräsentiert, so dass autorisierte Benutzer sie mit derselben relationalen Sprache abfragen können. Diese Regel ist sachlich bereits in Regel 1 enthalten.

5. Umfassende Datenmanipulationssprachenunterstützung

Ein relationales System kann verschiedene Sprachen unterstützen. Aber es muss eine Sprache existieren, für die gilt:

- Ihre Statements bestehen aus Zeichenketten in einer wohldefinierten Syntax;
- Mit ihr können folgende Funktionen erfüllt werden:
  - Datendefinition,
  - VIEW-Definition,
  - Datenmanipulation (interaktiv und durch Programm),
  - Integritätsregel-Definition,
  - Autorisierung,
  - Transaktionsgrenzen (begin, commit, rollback).

Solche Sprachen gibt es, etwa QUEL oder SQL, wobei sich letztere zu einem De-facto-Standard entwickelt hat. SQL ist allerdings nicht orthogonal und stark redundant. Außerdem hat es beim Programmieren einen anderen Umfang als im interaktiven Modus.

6. VIEW Datenbearbeitung

Alle VIEWS, die theoretisch manipuliert werden können, können durch das System manipuliert werden.

Diese Regel ist trennscharf. VIEW-Manipulationen wie DELETE, INSERT und UPDATE sind nicht in jedem der verbreiteten Produkte erlaubt. Dies basiert darauf, dass nicht alle möglichen Manipulationen korrekt sind, sondern nur solche, die sich eindeutig in eine zeitunabhängige Reihe von Manipulationen der zugrundeliegenden Basistabellen zerlegen lassen. Die Schwierigkeit, zwischen theoretisch korrekten und verfälschenden Manipulationen zu unterscheiden, wird von manchen Herstellern dadurch umgangen, dass VIEW-Manipulationen nur zugelassen werden, wenn der VIEW auf einer einzelnen Basistabelle beruht.

7. High-Level INSERT, UPDATE und DELETE

Die Möglichkeit, eine Tabelle oder einen VIEW als Operanden zu behandeln, existiert nicht nur für Lese-, sondern auch für Schreibzugriffe. In der Tat ist SQL eine gruppen- und keine satzorientierte Sprache. Dies wird besonders wichtig, wenn man verteilte DBMS betrachtet. Hier müsste pro Satz zwischen verschiedenen Rechnerknoten navigiert werden.

8. Physische Unabhängigkeit der Daten

Benutzeraktivitäten und Anwendungsprogramme müssen nicht geändert werden, wenn Veränderungen an der physischen Speicherung beziehungsweise der Zugriffsmethode vorgenommen werden.

Physische Datenunabhängigkeit im Sinn dieser Regel ist bei den heute käuflichen Produkten meist gegeben.

9. Logische Unabhängigkeit der Daten

Benutzeraktivitäten und Programme müssen nicht geändert werden, wenn informationserhaltende Veränderungen der Basistabellen irgendwelcher Art dies theoretisch erlauben.

Diese logische Datenunabhängigkeit unterstellt, dass die Anwendungen nicht direkt auf Basistabellen zugreifen. Dann würden deren Änderungen entsprechende Programmanpassungen notwendig machen, Man kann Applikationen statt dessen prinzipiell nur auf VIEWs zugreifen lassen. Nehmen wir an, Anwendung A verarbeite Tabelle T durch den VIEW T'. Wird T aus Performance-Überlegungen heraus auf zwei Tabellen T1, und T2 aufgeteilt, so muss lediglich der VIEW T' neu als Join von T1 und T2 definiert werden - ohne jede Änderung der Applikation läuft diese weiter, wenn das System Regel 6 (VIEW-Manipulation) unterstützt. Es gibt in der Praxis allerdings starke - und begründete - Vorbehalte gegenüber dem Ideal, durch VIEWs die Wartung der Anwendungen vermeiden zu können. Die meisten Änderungen an Basistabellen, die sich in der Produktion ergeben, machen nämlich auch die VIEWs unbrauchbar, die auf sie definiert sind.

10. Integritätsunabhängigkeit

Integritätsregeln, die einer bestimmten relationalen Datenbank angehören, müssen mit einer Subsprache definierbar und im Katalog des Systems ablegbar sein.

Dies ist wieder eine trennscharfe Regel. Es geht darum, spezifische, sich aus dem konkreten Inhalt der DB ergebende Integritätsregeln nicht in der Anwendung, sondern im DBMS zu verwalten. Es ist indes keineswegs so, dass alle kommerziellen RDBMS eine solche Prozedurspeicherung erlauben.

11. Unabhängigkeit bei verteilten Daten

Benutzeraktivitäten und Programme müssen nicht geändert werden, wenn

- eine Verteilung der Datenbasis auf mehrere Rechner vollzogen wird; und
- wenn die Daten in einem bestehenden verteilten System neu verteilt werden.

Die so charakterisierte Verteilungstransparenz ist ebenfalls trennscharf und äußerst wichtig. Wenn man davon ausgeht, dass das Konzept eines verteilten DBMS in der Zukunft eine entscheidende Rolle spielen wird, dann ist es von enormer Bedeutung, dass ihre Einführung die Investitionen in bestehende Anwendungssysteme schützt und nicht de-facto entwertet, da umfangreiche Anpassungen vorgenommen werden müssen.

12. Verbot der Umgehung von Integritätsregeln

Wenn ein relationales System eine satzorientierte Sprache unterstützt, kann diese nicht dazu benützt werden, die Integritätsregeln zu umgehen. Hier geht es der Sache nach um nicht-relationale Systeme, die mit einer SQL-Schnittstelle erweitert werden. In diesem Fall wäre es höchst riskant, wenn die mit der gruppenorientierten Sprache definierten Integritätszwänge sozusagen satzweise ignoriert werden könnten.

Insgesamt sind, wenn man das heutige Produktspektrum betrachtet, vor allem die Regeln 6, 10 und 11 bei der Einschätzung von Datenbanksystemen von großer Bedeutung. Die Regeln 1 bis 5 sowie 7 und 8 werden im Prinzip von den diversen SQL-Dialekten eingehalten. Regel 9 fällt mit Regel 6 zusammen, und die Regel 12 schließlich kennzeichnet einen Spezialfall, nämlich ein um ein relationales Interface erweitertes konventionelles DBMS.

ORACLE

Für das AIM System wurde das relationale Datenbank Management System der Firma ORACLE zur Aufnahme des Faktenwissens gewählt. Die Argumente die zusätzlich zu den Vor- und Nachteilen von relationalen DBMS für eine Verwendung des ORACLE RDBMS sprachen sind im folgenden aufgelistet:

- Eine ältere Version des ORACLE RDBMS war bereits am Institut vorhanden
  und ein (Update relativ preiswert)
- Relationale DBMS sind Industriestandard
- ORACLE RDBMS wird auf fast allen Betriebsystemen und Rechnerplattformen unterstützt
  (einschließlich Apple Macintosh)
- Ein Programmierinterface von ORACLE zu MacProlog ist vorhanden
- Ein Programmierinterface zwischen Hypertalk und ORACLE ist vorhanden
- Kompatibilität mit anderen Datenbanken (ODBC, Import, Export, etc.)

Im Bezug auf die Codd Regeln schneidet ORACLE im Vergleich mit anderen DBMS noch relativ gut ab, wie die folgende Auflistung zeigt:

- Uneingeschränkte Erfüllung der Regeln 1, 3, 4, 6, 9, 12
- Regel 2: ORACLE erlaubt aus Effektivitätsgründen doppelte Zeilen, die Einhaltung der Regel liegt somit in der Verantwortung des Programmierers
- Teilweise Erfüllung der Regel 5: SQL unterstützt DDL, VDL, DML, Benutzerautorisierung und Transaktionen; Integritätsbedingungen werden nicht unterstützt (Ausnahme: Entity Integrität über NOT NULL); SQL2 beinhaltet Unterstützung für referentielle Integrität über Schlüsseldefinitionen
- Regel 7: UPDATE und DELETE sind mengenbasiert; INSERT immer nur Zeile für Zeile
- Gute Erfüllung von Regel 8: INDIZES und CLUSTER beeinflussen SELECT Anweisungen nicht
- Regel 10: Nur Unterstützung von Entity Integrität
- Regel 11: In der vorliegenden Version RDBMS V6.0.36.1.0 keine verteilten Daten, daher auch keine Probleme

2.2.2 Grundlagen der Theorie relationaler Datenbanken

Für das Verständnis der gewählten Struktur der Datenbank ist es unerlässlich einige Begriffe der Datenbanktheorie zu klären und einige Modellvorstellungen vorzustellen. Auch soll hier kurz auf die verschiedenen Datenbankarten eingegangen werden, die in der heutigen EDV Verwendung finden. Aufgrund der in diesem Kapitel erläuterten Grundlagen werden die Entscheidungen die während der Entwicklung des AIM Informationssystems zu treffen waren plausibel.

Datenmodellierung

In der Datenmodellierung wird versucht, die Realität oder einen Ausschnitt davon im Computer abzubilden. Hierzu ist es notwendig, in der Realität vorhandene Strukturen zu erkennen und als Modellschemata für die elektronische Speicherung zu verwenden. Je komplexer die Realität, desto komplexer muss auch die Struktur sein, um eine Erfassung aller geforderten Daten ohne logische Verletzungen der intuitiven Strukturierung vornehmen zu können. Im Zusammenhang mit der Datenmodellierung werden folgende Definitionen erwähnt /47/:

- die **Diskurswelt** ist der Ausschnitt aus der Realität, der in der Datenbank erfasst werden soll,
- die **Datenbasis** ist das Modell der Diskurswelt. Sie besteht aus einer Menge von Objekten, die untereinander in Beziehung stehen. Jedes Objekt wird durch eine Kombination von Werten repräsentiert.
- das **Datenschema** oder **Schema** beschreibt die konstante Struktur der Datenbasis. Es besteht aus einer Menge von Objekttypen (Klassen), die durch Attribute beschrieben werden und untereinander in Beziehung stehen.
- das **Datenmodell** ist ein Gestaltungsrahmen für die Aufstellung von Schemata. Dieser Rahmen legt die Arten von Objekttypen und Beziehungen fest.

Die Aufgabe der Datenmodellierung ist die Aufstellung des Datenschemas. Für die konkrete Aufgabe der Aufstellung des Datenschemas der AIM Datenbank wurden das Entity Relationship Modell (ERM) /6/ und teilweise die von der Universität Bamberg eingeführte Erweiterung, das strukturierte Entity Relationship Modell (SERM) /47/, verwendet. Die historischen Datenmodelle (Relationenmodell, Netzwerkmodell und hierarchisches Modell) werden an dieser Stelle nicht weiter ausgeführt.

Schlüssel und Indizes

Die Begriffe Schlüssel und Indizes werden oft verwechselt. Da sie aber für ein Verständnis der Datenstrukturen der AIM Datenbank notwendig sind, soll hier eine kurze Erläuterung mit Beispielen aus der AIM Datenbank gegeben werden. Die Definitionen sind den ORACLE Handbüchern entnommen /30/.

- **Indizes (Indexes)** werden benutzt, um den Zugriff auf einzelne Zeilen einer Tabelle zu beschleunigen. Sie sind Bestandteil der Datenbank und werden vom Benutzer definiert und angelegt. Ein Index kann mehrere Spalten der Tabelle umfassen. Das Anlegen eines "UNIQUE" Index stellt sicher, dass innerhalb einer Tabelle keine doppelten Einträge möglich sind.

  Beispiel: Der UNIQUE Index `IxLauncher` auf der Spalte `Launcher_ID` der Tabelle `Launcher` stellt sicher, dass keine doppelten IDs vorkommen. Zusätzlich wird der Zugriff auf die Zeilen der Tabelle `Launcher` über die `Launcher_ID` beschleunigt. Dem gleichen Zweck dient auch der UNIQUE Index `IxLauncher_Name` auf der Spalte `Launcher_Name` der Tabelle `Launcher`, der sicherstellt, dass keine doppelten Trägerraketennamen eingetragen werden können und die Suchfunktion nach bestimmten Trägernamen beschleunigt werden.

- **Schlüssel (keys)** sind ein logisches Konzept aus der Datenbanktheorie. Ein **Kandidatenschlüssel (candidate key)** kennzeichnet eine Spalte (oder eine Gruppe von Spalten) einer Tabelle. Ein **Primärschlüssel** (primary key) ist eine Spalte (oder eine Kombination von Spalten) einer Tabelle, der benutzt werden kann, um jede beliebige Zeile einer Tabelle eindeutig zu identifizieren. Ein Primärschlüssel ist einer der Kandidatenschlüssel einer Tabelle.

  Beispiel: Die Spalte `Launcher_ID` der Tabelle `Launcher` dient als Primärschlüssel, für die eindeutige Identifizierung der Zeilen der Tabelle.

  Enthält eine Tabelle weitere Spalten, über welche die Zeilen der Tabelle eindeutig identifiziert werden können, dann spricht man in diesem Fall auch von **Sekundärschlüsseln**.

  Beispiel: In der Tabelle `Launcher` wäre die Spalte `Launcher_Name` ein solcher Sekundärschlüssel.

  Ein **Fremdschlüssel (foreign key)** ist eine Spalte (oder eine Gruppe von Spalten) in einer Tabelle, die mit einem Primärschlüssel einer anderen Tabelle korrespondieren. Fremdschlüssel werden benutzt, um Datenobjekte aus mehreren Tabellen zu kombinieren.

  Beispiel: In der Tabelle `Launcher_Stage` sind die Spalten `Launcher_ID` und `Stage_ID` Fremdschlüssel für die Identifizierung der Zeilen der Tabellen `Launcher` bzw. `Stage`. Sie bilden zusammen mit der Spalte `Launcher_StageBlockNo` einen UNIQUE Index `IxLauncher_Stage` um sicherzustellen, dass ein Träger nicht mehrere verschiedene Stufen als einen Block zugeordnet bekommt.

Ein Index oder Schlüssel, der sich aus mehreren Spalten einer Tabelle zusammensetzt, wird auch zusammengesetzter Index (concatenated index) bzw. zusammengesetzter Schlüssel (concatenated key) genannt.

Beziehungen/Relationen

Wie im vorherigen Absatz erwähnt, werden Relationen zwischen Objekten über Fremdschlüssel hergestellt. Drei Arten von Relationen kommen in der AIM Datenbank vor:

- Echte Relationen die eine einfache Beziehung zwischen zwei Objekttypen herstellen.
  Beispiel: Die Tabelle `Reference_Folder` enthält nur die jeweiligen Fremdschlüssel `Reference_ID` und `Folder_ID`
- Relationen die keine Beziehung zwischen Objekttypen herstellen, sondern nur einen Zusammenhang zwischen komplexen Attributen eines Objekttyps.
  Beispiel: In der Tabelle `Stage_Mass` werden die Massendaten der Stufe gespeichert. Die Spalte `Stage_MassType` enthält die Beschreibung des Attributs (z.B. "Tank Mass", "Total Mass", "Dry Mass", etc.) und `Stage_MassValue` den Zahlenwert. Als Einheit wird immer "kg" angenommen. Die Tabelle enthält keine Fremdschlüssel außer `Stage_ID`
- Fremdschlüssel-Beziehungen die sogenannte Komponentenobjekte modellieren.
  Beispiel: Tabelle `Mission`. Ein Attribut des Objekttyps `Mission` ist wiederum ein Objekttyp (`Launch`), repräsentiert durch den Fremdschlüssel `Launch_ID`.

Außerdem gibt es noch eine zusätzliche, für objektorientierte Datenbanken relativ wichtige Beziehung, die aber in der AIM Datenbank nur an einer Stelle Verwendung findet:

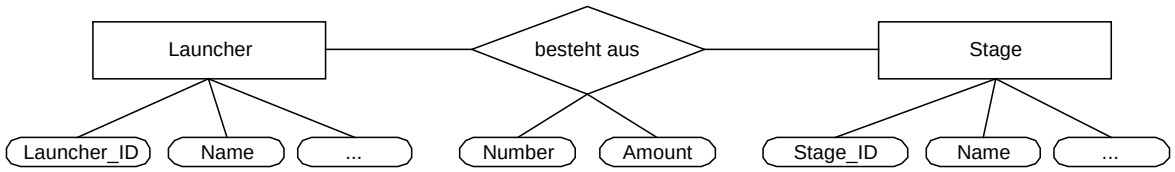
- Relationen, die eine "is a"-Beziehung zwischen zwei Objekttypen herstellen
  Beispiel: Die Tabelle `Propellant` enthält teilweise "is-a"-Beziehungen mit der Tabelle `Substance`. `Propellant` enthält Substanzen (oder Kombinationen von Substanzen), die auch Treibstoffe sind. Treibstoffe die nur aus einer Substanz bestehen stellen somit eine is-a Beziehung zwischen `Substance` und `Propellant` her. In der Tabelle `Propellant_Substance` wird diese Relation gespeichert. Ansonsten gibt es in der AIM Datenbank keine weiteren "is a" – Beziehungen.

Entity-Relationship Modell

Die in der realen Welt vorkommenden Objekte werden im ERM als Gegenstände (Entity) repräsentiert. Die Verknüpfungen zwischen zwei oder mehr Gegenständen wird als Beziehung (Relationship) bezeichnet. Gleichartige Gegenstände werden zu einem Gegenstands-Objekttyp, gleichartige Beziehungen zu einem Beziehungs-Objekttyp zusammengefasst. Beide Objekttypen werden durch eine bestimmte Anzahl Attribute beschrieben. Ein Gegenstands-Objekttyp muss Attribute, ein Beziehungs-Objekttyp kann Attribute zugeordnet haben.

In der AIM Datenbank stellen zum Beispiel die Tabellen `Launcher` und `Stage` Gegenstands-Objekttypen dar. `Ariane 44LP` ist ein Objekt des Objekttyps `Launcher`, `L 220` ist ein Objekt des Objekttyps `Stage`. Die Tabelle `Launcher_Stage` steht für einen Beziehungs-Objekttyp der Art `"besteht aus"` und der Eintrag `Ariane 44 LP,1,1, L 220` ist eine Instanz dieses Objekttyps. Die Spalten der Tabellen stellen die Attribute dar.

Objekte in relationalen Datenbanken werden in der Regel durch ganze Zahlen eindeutig identifiziert. In Abb. 2-1 werden die Objekte des Gegenstands-Objekttyp `Launcher` durch die Spalte `Launcher_ID` und die des Objekttyps `Stage` durch die Spalte `Stage_ID` eindeutig identifiziert.



Lcher_ID	Name	…
8	Ariane 44 LP	…
…	…	…

Lcher_ID	Stage_ID	Amount	Number
8	10	1	1
…	…	…	…

Stage_ID	Name	…
10	L 220	…
…	…	…

Abb. 2-1 Objekte und Objekttypen

Eine Relation stellt die Verbindung zwischen zwei oder mehr Gegenständen her. In Abb. 2-1 stellt die Relation "besteht aus" (repräsentiert durch die Tabelle `Launcher_Stage`) die Verbindung zwischen den Gegenständen `Launcher` und `Stage` her. Bei einer reinen Relation enthält die Relationship Tabelle nur die beiden Fremdschlüssel (`Launcher_ID`, `Stage_ID`). In einer Entity-Relationship Tabelle wie in dem gezeigten Beispiel sind zusätzliche Informationen in zusätzlichen Spalten gespeichert. In diesem Fall die Anzahl der Stufen (Amount) und die Reihenfolge der Stufen ausgedrückt durch die Stufennummer (Number).

Komplexität

Die Komplexität einer Beziehung beschreibt das Verhältnis der Gegenstände zueinander. Bei Verwendung der (1,M,N)-Notation können folgende Komplexitäten vorkommen:

- 1:1
- 1:N
- M:N

1:1 hieße in dem im vorigen Absatz verwendeten Beispiel: Ein Träger kann genau eine Stufe besitzen und eine Stufe kann zu genau einem Träger gehören.

1:N hieße ein Träger kann beliebig viele Stufen besitzen und eine Stufe kann aber nur zu genau einem Träger gehören.

M:N heißt dann: Ein Träger kann beliebig viele Stufen besitzen und eine Stufe kann zu beliebig vielen Trägern gehören.

Die obige Notation ist mehrdeutig. Bei einer 1:N Beziehung wie im vorigen Absatz zwischen zwei Objekttypen `Launcher` und `Stage` wird nichts darüber ausgesagt, ob jedem Träger mindestens eine Stufe zugeordnet sein muss. Umgekehrt ist nicht ersichtlich, ob sich jede Stufe auf genau einen Träger bezieht oder ob Stufen ohne Träger überhaupt zulässig sind. Aus diesem Grund wird in /47/ eine noch weitergehende Definition der Komplexität von Beziehungen gegeben. Durch Einführung eines Existenzkriteriums ergeben sich folgende Komplexitäten:

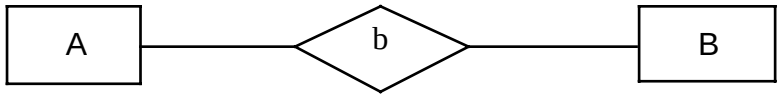
- 0:1
- 0:*
- 1:1
- 1:*

In Abb. 2-2 sind die Beziehungen zwischen den Objekttypen A und B in min,max Notation /45/ ($0 \leq \min \leq 1 \leq \max \leq *$; $* \equiv$ "beliebig viele") aufgeführt. Hierbei ist A der originäre Objekttyp und B der abhängige Objekttyp. Der Komplexitätsgrad comp(A,B) gibt an, mit wie vielen Datenobjekten des Typs B ein Datenobjekt des Typs A minimal in Beziehung stehen muss und maximal in Beziehung stehen kann.

Aus den Eckwerten des Komplexitätsgrades (0,1), (0,*), (1,1) und (1,*) sind zwei Formen von Existenzabhängigkeiten ableitbar:

- Einseitige Existenzabhängigkeit (B hängt von A ab):
  comp(A,B) = (0,1) oder comp(A,B) = (0,*)
- Wechselseitige Existenzabhängigkeit (B und A sind wechselseitig abhängig): comp(A,B) = (1,1) oder comp(A,B) = (1,*)

Eine Beziehung b(A,B) wird durch zwei Komplexitätsgrade comp(A,b) und comp(B,b) beschrieben. In Abb. 2-2 sind die (1,M,N)-Notation und die (min,max)-Notation gegenübergestellt.



b(A,B)	comp(A,b)	comp(B,b)
1:1	(0,1) oder (1,1)	(0,1) oder (1,1)
1:N	(0,*) oder (1,*)	(0,1) oder (1,1)
N:1	(0,1) oder (1,1)	(0,*) oder (1,*)
N:M	(0,*) oder (1,*)	(0,*) oder (1,*)

Abb. 2-2 Komplexität der Beziehungen /47/

Normalformen

Um die Definition der verschiedenen Normalformen zu verstehen, müssen zunächst einige Begriffe geklärt werden. In /17/ und /18/ werden die verschiedenen Abhängigkeiten folgendermaßen erläutert:

- Funktionale Abhängigkeit (functional dependency, FD):

  Eine Funktionale Abhängigkeit gilt dann innerhalb einer Relation zwischen Attributmengen X und Y, wenn in jedem Tupel der Relation der Attributwert unter den X-Komponenten den Attributwert unter den Y-Komponenten festlegt. Unterscheiden sich also zwei Tupel in den X-Attributen nicht, so haben sie auch gleiche Werte für alle Y-Attribute. Man schreibt X→Y /17/.

  Eine Menge von Attributen sind funktional abhängig von einer anderen Menge von Attributen, wenn der oder die Werte der letzteren Menge den oder die Werte der ursprünglichen Menge bestimmen /18/.

- Mehrwertige Abhängigkeit (multi-valued dependency, MVD):

  Eine Mehrwertige Abhängigkeit beschreibt die Zuordnung einer Menge von Attributwerten zu einem anderen Attributwert. Sie besagt, dass innerhalb einer Relation r einem Attributwert von X einer Menge von Y-Werten zugeordnet wird, unabhängig von den Werten der restlichen Attribute von r. Man schreibt X→→Y.

Da in der aktuellen Version AIM Datenbank als Folge der Normalisierung nicht alle Abhängigkeiten vorkommen, soll hier auf das in Tab. 2-1 dargestellte Beispiel aus der Entwurfsphase der AIM Datenbank zurückgegriffen werden. In der Tabelle gilt die funktionale Abhängigkeit

`ReportNumber`→`Title`, `PublicationDate`,

da zwei in der Reportnummer übereinstimmende Quellen auch in den Attributen Titel und Erscheinungsdatum übereinstimmen müssen.

Eine funktionale Abhängigkeit

`ReportNumber`→`Author`, `Keyword`,

gilt dagegen nicht, da eine Quelle unterschiedliche Stichworte und Autoren haben kann. Dagegen gelten die mehrwertigen Abhängigkeiten

`ReportNumber`→→`Author` und `ReportNumber`→→`Keyword`,

da der jeder Reportnummer zugeordnete Autor einmal in einem Tupel mit jedem der Reportnummer zugeordneten Stichwort auftauchen muss, damit er wirklich unabhängig davon ist. Ein Schlüssel kann ebenfalls über funktionale Abhängigkeiten definiert werden. Ein Schlüssel ist für ein Relationenschema R eine funktionale Abhängigkeit X→R, wenn X minimal ist.

Während die oben genannten Abhängigkeiten beim formalen Datenbankentwurf eine relativ große Rolle spielen, werden die im weiteren genannten meistens vernachlässigt. Daher sollen sie hier nur der Vollständigkeit halber genannt werden:

- Verbundabhängigkeit (join dependency, JD)
  Von Verbundabhängigkeit spricht man, wenn man ein Relationenschema ohne Informationsverlust in Schemata $R_1,\ldots, R_p$ auftrennen kann. Die Vereinigung aller R_i soll dabei R ergeben. Man schreibt: $\triangleright\triangleleft[R_1,\ldots, R_p]$
- Inklusionsabhängigkeit (inclusion dependency, IND)
  Eine Inklusionsabhängigkeit ist gegeben wenn die X-Werte einer Relation $r(R_1)$ auch als Y-Werte einer Relation $r(R_2)$ vorkommen. Man schreibt: $R_1[X]\subseteq R_2[Y]$

An der Struktur der in einem Relationenschema geltenden Abhängigkeiten erkennt man, ob in dieser Relation redundante Daten gespeichert werden können. Zwei solcher bösartigen Strukturen sollen hier kurz erläutert werden:

- Transitive Abhängigkeit
  Bestimmt ein Schlüssel K eine Attributmenge X funktional, und bestimmt die Attributmenge X aber auch eine Attributmenge Y des gleichen Schemas, dann liegt eine transitive Abhängigkeit vor. Man schreibt: K→X→Y.
  In /17/ wird formal definiert: Sei R Relationenschema, X⊆R, F FD-Menge über R. Ein A∈R heißt transitiv abhängig von X bezüglich F genau dann, wenn es ein Y⊆R gibt mit X→Y, Y↛X, Y→A, A∉XY.
  Eine transitive Abhängigkeit besteht dann, wenn einige Attribute funktional abhängig von anderen Attributen sind, welche wiederum funktional abhängig vom Primärschlüssel sind. Das Vorhandensein einer transitiven Abhängigkeit deutet an, dass sich eine zusätzliche Tabelle in der zu Normalisierenden "versteckt" /18/.
- Nichttriviale Abhängigkeit
  Eine MVD X→→Y ist nicht-trivial, wenn außer X und Y noch weitere Attribute im zugehörigen Relationenschema enthalten sind.
- In /17/ wird formal definiert: Sei R Relationenschema, X,Y⊆R, M MVD-Menge über R. Eine MVD X→Y heißt trivial genau dann, wenn Y⊆X oder X∪Y=R ist.

In dem in Tab. 2-1 gezeigten Beispiel versteckt sich eine transitive Abhängigkeit. Die Information der Vornamen der Autoren ist redundant gespeichert. Wenn man davon ausgeht, dass der Nachname den Vornamen eindeutig identifiziert, dann liegt eine transitive Abhängigkeit `ReportNumber→Author→AuthorFirstname` vor. Außerdem gelten die MVDs `ReportNumber→Author` und `ReportNumber→Keyword`. Da aber noch weitere Attribute im Relationenschema vorhanden sind, ist keine dieser MVDs trivial.

Das Beseitigen solcher "bösartigen" Datenstrukturen nennt man Normalisierung. Im Prinzip geht es dabei darum, verschiedene Ausprägungen funktionaler Abhängigkeiten zu entfernen. Dieser Prozess sollte jedoch nicht isoliert von anderen Entwurfsprinzipien einer Datenbank gesehen werden (s.a. Abschnitt: Denormalisierung). Bei dem Prozess der Normalisierung werden verschiedene Stufen durchlaufen die Normalformen genannt werden.

Report Number	Title	Author	Author Firstname	Keyword	Publication Date
IAF-90-241	Some Recent Developments In Liquid Propulsion Systems In ISRO	Muthunayagam	A.E.	Liquid Propellant Engines	06-OCT-90
IAF-90-241	Some Recent Developments In Liquid Propulsion Systems In ISRO	Muthunayagam	A.E.	Ground Tests	06-OCT-90
IAF-90-241	Some Recent Developments In Liquid Propulsion Systems In ISRO	Muthunayagam	A.E.	Subsystem	06-OCT-90
IAF-90-243	Oxidizer Size Effect On Dynamic Combustion Of AP/HTPB Composite Propellants	Osborn	G.H.	Solid Propellant Combustion	06-OCT-90
IAF-90-243	Oxidizer Size Effect On Dynamic Combustion Of AP/HTPB Composite Propellants	Jian	Z.Q.	Equations	06-OCT-90
IAF-90-243	Oxidizer Size Effect On Dynamic Combustion Of AP/HTPB Composite Propellants	Osborn	G.H.	Equations	06-OCT-90
IAF-90-243	Oxidizer Size Effect On Dynamic Combustion Of AP/HTPB Composite Propellants	Jian	Z.Q.	Solid Propellant Combustion	06-OCT-90

Tab. 2-1 Die Tabelle `Reference` in der Entwurfsphase der AIM Datenbank

Nachdem die wichtigsten Begriffe geklärt sind, können nun die Definitionen für die verschiedenen Normalformen für Datenschemata erläutert werden. Bei den einzelnen Erläuterungen wird versucht sowohl eine allgemein verständliche als auch eine im Sinne der Datenbanktheorie formale Erklärung anzugeben. Die allgemeinen Definitionen sind den Quellen /9/ und /18/ entnommen, die formalen Definitionen der Quelle /17/.

1. Normalform (1NF)

Eine Datenstruktur befindet sich in 1. Normalform, wenn die folgenden Bedingungen erfüllt sind:

- Jedes Datenelement einer Datengruppe muss einen eindeutigen Namen besitzen bzw. kein Datenelement darf mehrfach in einer Gruppe vorkommen.
- Jedes Datenelement muss atomar sein, das heißt, es darf nicht in kleinere Einheiten zerlegbar sein.
- Jede Datengruppe besitzt einen Primärschlüssel
- Assoziationen werden lediglich durch Schlüssel und nicht durch absolute Zeiger vermittelt.

Die Bedingung der Atomarität ist bei relationalen Datenbanken von vornherein erfüllt. Dagegen kann das Vorhandensein von MVDs manchmal sehr nützlich sein und dem intuitiven Verständnis des

Benutzers entgegenkommen. Daher wird bei der aktuellen Forschung im Datenbankbereich daran gearbeitet die erste Normalform zu streichen /18/.

2. Normalform (2NF)

Eine Datenstruktur befindet sich in 2. Normalform, wenn die folgenden Bedingungen erfüllt sind:

- Die Datengruppe besitzt die erste Normalform
- Jedes Datenelement, das nicht zum Primärschlüssel gehört, ist voll funktional abhängig vom Primärschlüssel

Die ersten beiden Normalformen sind hier nur der Vollständigkeit halber genannt worden. Für den Datenbankentwurf sind nur die im folgenden erläuterten Normalformen von Belang, da diese einigermaßen konsistente Datenstrukturen versprechen.

3. Normalform (3NF)

Eine Datenstruktur befindet sich in 3. Normalform, wenn die folgenden Bedingungen erfüllt sind:

- Die Datengruppe besitzt die zweite Normalform
- Jedes Datenelement, das nicht zu einem Kandidatenschlüssel der Datengruppe gehört, hängt nicht-transitiv von jedem dieser Kandidatenschlüssel ab.

Die formale Definition lautet: Ein erweitertes Relationenschema $R=(R,K)$ ist in dritter Normalform bezüglich einer FD-Menge F genau dann, wenn

$\nexists A \in R$: A ist kein Primärattribut in R
$\wedge$ A transitiv abhängig von einem Schlüssel $K \in K$ bezüglich F.

Boyce-Codd-Normalform (BCNF)

Bei der BCNF handelt es sich um eine Sonderform, aber noch nicht um die 4. Normalform. Man kann zeigen, das „S ist in BCNF$\Rightarrow$S ist in 3NF“ gilt, aber nicht umgekehrt. Die formale Definition lautet: Ein erweitertes Relationenschema $R=(R,K)$ ist bezüglich F in Boyce-Codd-Normalform genau dann, wenn

$\nexists A \in R$: A transitiv abhängig von einem Schlüssel $K \in K$ bezüglich F.

4. Normalform (4NF)

Eine Datenstruktur befindet sich in 4. Normalform, wenn die folgenden Bedingungen erfüllt sind:

- Die Datengruppe besitzt die dritte Normalform
- Die Datengruppe besteht aus einer Menge wechselseitig unabhängiger Datenelemente, die alle vom Primärschlüssel der Datengruppe funktional abhängig sind.

Die formale Definition lautet: Sei R Relationenschema, $X,Y \subseteq R$, M MVD-Menge über R. Ein erweitertes $R=(R,K)$ ist in 4. Normalform bezüglich M genau dann, wenn

$\forall (X \rightarrow\rightarrow Y) \in M$: $X \rightarrow\rightarrow Y$ ist trivial $\vee$ $X \supseteq K$ für ein $K \in K$.

In einigen Quellen wird noch eine fünfte Normalform erwähnt, welche aber nur theoretischen Charakter hat und deshalb hier nicht näher erläutert wird.

Denormalisierung

Beim Datenbankentwurf wird darauf geachtet alle Relationen in 4NF zu bringen, um ein Höchstmass an Redundanzfreiheit zu gewährleisten. Dies kann nicht immer erreicht werden, da ab einem gewissen Zerlegungsgrad die Auffassungsgabe des Benutzers erschöpft ist. Außerdem erzeugt man durch die Anwendung der Normalformregeln mitunter sehr viele Einzeltabellen, die Abfragen an die Datenbank aus logischen Gesichtspunkten und aus Gründen der Performance extrem erschweren können.

Der Prozess der Normalisierung dient, wie bereits gesagt, der Verbesserung des logischen Datenbankentwurfs und verhindert Redundanzen und reduziert die Gefahr von Anomalien bei der Überarbeitung bestehender Daten. Dies bedeutet aber zwangsläufig die Erzeugung zahlreicher Tabellen und die Verteilung der Informationen auf mehrere Tabellen. Denormalisierung bedeutet eine Umkehr dieses Prozesses, um eine Steigerung der Verarbeitungsgeschwindigkeit zu erreichen. Mehr Tabellen bedeuten mehr JOINs in SQL-Abfragen; mehr JOINs bedeuten schlechtere Performance. Es wird empfohlen nicht mehr als drei Tabellen in einer SQL-Abfrage per JOIN zu verbinden /18/. Als weiterer Grund für eine Denormalisierung kann angeführt werden, dass manchen Fällen die Art wie die in der Datenbank gespeicherten Informationen benutzt werden, eine strikte Normalisierung unnötig macht.

2.2.3 Organisation von relationalen Datenbanken

Heutzutage sucht der Raumfahrtingenieur und somit auch der Student der Raumfahrttechnik seine für eine bestimmte Aufgabe benötigten Informationen nicht mehr ausschließlich in Bibliotheken. Zeitschriften und Bücher, die traditionellen Quellen für Informationen, sind teilweise durch die elektronischen Medien oder computerbasierte Datenbanken ersetzt worden.

Selbst beim Zugriff auf Bibliotheksbestände haben sich Computer als Suchmaschinen durchgesetzt. Die folgenden Abschnitte beschränken sich auf die EDV-basierten Informationssysteme und deren Zugriffsmöglichkeiten.

Einzelplatzlösungen

Unter Einzelplatzlösungen versteht man isolierte Computer (isoliert heißt hier: nicht vernetzt) die von einer einzigen Person zur selben Zeit, oder von mehreren Personen nacheinander abwechselnd benutzt werden. Der gesamte Datenbestand existiert lokal auf eben diesem Rechner und kann sehr schnell und effizient abgefragt und geändert oder ergänzt werden. Allerdings ist der Platz in der Regel beschränkt und die angebotene Informationsmenge richtet sich nach dem lokal verfügbaren Speicherangebot.

Die Geschwindigkeit der Informationsverarbeitung wird ebenfalls nur von der Leistungsfähigkeit des lokalen Rechners bestimmt. Die Geschwindigkeit mit der Informationen gefunden werden, wird durch die Menge der vorhandenen Informationen und durch die Leistung des lokalen Rechners bestimmt.

Am Institut für Luft- und Raumfahrt der TU Berlin existieren verschiedene Einzelplatzlösungen. Zum Beispiel eine alte Version der AIM Datenbank basierend auf dem Apple Macintosh Programm HyperCard. Diese alten HyperCard Stacks bilden Einzelplatzlösungen für die Bereiche „Triebwerke“, „Träger“, „Kosten“ und „Raketenstarts“. Dann existiert eine umfangreiche Zitatensammlung, die über eine Baumstruktur durchsucht werden kann. Unter den neueren Entwicklungen ist eine Treibstoffdatenbank unter MS Excel hervorzuheben.

Client/Server Architekturen

Unter Client/Server Architektur versteht man die Aufteilung der Aufgaben bei der Informationssuche auf verschiedenen Computer, die untereinander durch ein Netzwerk verbunden sind. Grundsätzlich sind der Ort, an dem die Informationen vorrätig gehalten werden und der Ort an dem die Benutzerschnittstelle (user interface) sich befindet unterschiedlich. Man benutzt sehr schnelle und leistungsfähige Systeme mit sehr großen Massenspeichermedien für die Datenbankfunktionen. Auf diesen Rechnern läuft ein Programm, die sogenannte Database Engine oder Database Management System. Bei relationalen Datenbanken, wie z.B. bei der am Institut verwendeten ORACLE Datenbank, wird diese Komponente RDBMS = Relational Database Management System genannt.

Weiterhin kann der Datenvorrat auf mehrere Rechner verteilt sein, die durch ein Netzwerk verbunden sind. Man spricht dann von *Verteilten Datenbanken* oder *Verteilten Systemen*. Die einzelnen RDBMS (jeder Rechner in solch einem verteilten System hat ein eigenes RDBMS laufen) sprechen sich untereinander ab, um die Datenkonsistenz bei Mehrfachzugriffen zu gewährleisten und die Zugriffsrechte untereinander abzustimmen.

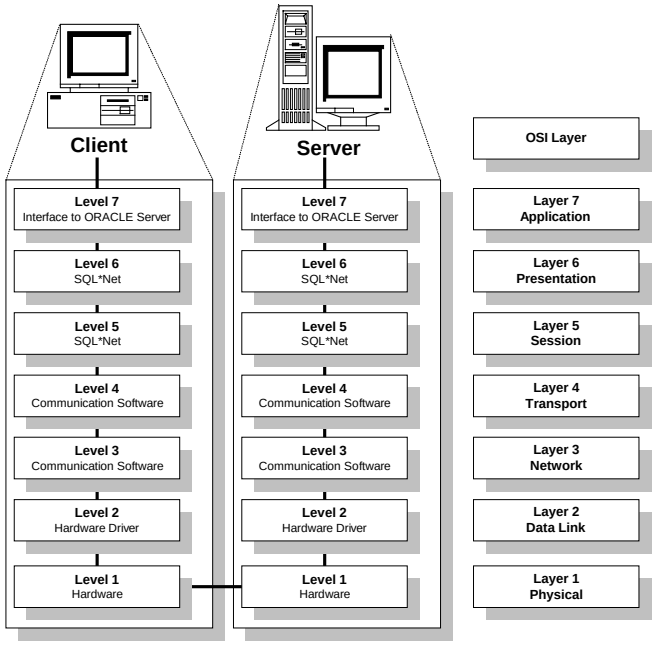


Abb. 2-3 Netzwerkkomponenten bei Client/Server Architektur /30/

Die Abb. 2-3 zeigt die Verteilung der ORACLE Softwarekomponenten, die in der AIM Datenbank zum Einsatz kommen, auf die jeweiligen OSI Schichten bei einer Client/Server Lösung. Bei Einzelplatzlösungen würden die Schichten 1 bis 6 wegfallen. Die Softwarekomponenten sind hardware- und betriebssystemabhängig. Da die Firma ORACLE jedoch für alle gängige Betriebssysteme Komponenten zur Verfügung stellt und die jeweiligen Hardwarehersteller für nahezu alle Betriebssysteme die zugehörigen Hardwaretreiber anbieten ist eine beliebige Mischung von Hardware und Betriebssystemen vorstellbar. So wird beim AIM Informationssystem als Datenbankserver ein Apple Macintosh verwendet. Die Client Computer sind sowohl Unix Workstations, IBM PC-Kompatible und Apple Computer. Genauso gut könnte man sich einen Unix-Rechner oder einen PC-Kompatiblen als Datenbankserver vorstellen. Allerdings müsste man von ORACLE die jeweilige plattformspezifische Server Software erwerben. Da bei der Einrichtung der AIM Datenbank bereits ein ORACLE Server für den Macintosh vorhanden war, wurde entschieden auf dieser Plattform aufzubauen. Die entsprechende Client Software für Apple und PC-Kompatible wurde zusätzlich erworben.

2.2.4 Darstellung von Informationen als Ergebnis von Abfragen

Wie bereits erwähnt, beeinflusst die Struktur einer relationalen Datenbank die Geschwindigkeit und die Benutzerfreundlichkeit einer Datenbankabfrage zum Teil erheblich. Je komplizierter die Tabellenstruktur desto länger können die Antwortzeiten auf solche Anfragen werden. Das hängt im starken Maß auch von den bei einer einzigen Abfrage beteiligten Tabellen ab. Besteht eine Abfrage aus sehr vielen Relationen (im ungünstigsten Fall aus MVDs) und ist die Anzahl der Reihen in den beteiligten Tabellen ebenfalls sehr groß, so können Antwortzeiten von mehreren Minuten die Folge sein. Dies hängt unter anderem auch von der Server Performance ab. Der im AIM System verwendete Apple Macintosh IIci gehört nicht zu den schnellsten und modernsten Rechnern. So kam es zum Beispiel bei einigen Abfragen zu Wartezeiten von 20-30 Minuten.

Im Bereich der relationalen Datenbanken hat sich die Abfragesprache SQL (Structured Query Language) als Standard etabliert. Eine Anfrage an ein RDBMS besteht letztendlich immer aus einem oder mehreren SQL-Kommandos. Selbst wenn zusätzliche Client Software ein benutzerfreundliches Interface zur Verfügung stellt werden alle Eingaben des Benutzers letztendlich in SQL-Kommandos umgesetzt und an das RDBMS weitergeleitet. Die Antwort des RDBMS besteht immer aus einer Tabelle mit 1 bis n Spalten und einer Anzahl von 1 bis n Zeilen. Diese Antwort kann ebenfalls durch zusätzliche Software weiter aufbereitet und in eine für den Benutzer bequemere Form gebracht werden. Die Client Software

oder der Benutzer muss jedoch Kenntnisse über die Tabellen- bzw. Datenstruktur der jeweiligen Datenbank besitzen. Es müssen alle Tabellennamen und deren Spalten bekannt sein. Der Ort an dem eine Information zu finden ist, also im einfachsten Fall die Tabelle und die Spalte, muss mit einem SQL-Kommando abgefragt werden. Bei schwierigeren Fällen, wie z.B. über mehrere Tabellen verteilte Informationen, müssen sowohl die Tabellen- und Spaltennamen der eigentlichen Informationsträgertabellen, als auch sämtliche Relationaltabellen für die JOIN Verknüpfung bekannt sein. Außerdem muss im SQL-Kommando die Verknüpfung der Tabellen in sinnvoller Weise bewerkstelligt werden. Auf die im AIM System verwirklichten Möglichkeiten der Datenabfrage und die damit verbundenen Darstellungsformen der Informationen wird in Kapitel 1 ausführlich eingegangen.

3 Künstliche Intelligenz und Expertensysteme

Dieser Abschnitt wird kurz auf die verschiedenen Gebiete der Künstlichen Intelligenz eingehen, um eine Einordnung des AIM Informations- und Expertensystems zu ermöglichen. Außerdem sollen die für diese Arbeit wichtigen Aspekte von Expertensystemen erläutert werden, um darlegen zu können, wie wichtige Merkmale von Expertensystemen beim AIM Informationssystem integriert wurden. Hierbei lässt es sich leider nicht vollständig vermeiden den Inhalt der vielen Bücher, die über Expertensysteme und Künstliche Intelligenz geschrieben wurden zu wiederholen. Dennoch sind die im folgenden aufgeführten Informationen wichtig, um eine Einordnung des AIM Systems, der angewandten Methoden der Künstlichen Intelligenz und des Aufbaus der Expertensystemkomponenten von AIM transparent zu machen.

3.1 Künstliche Intelligenz

Um eine Einordnung des AIM Informationssystems vornehmen zu können, muss zunächst ein Überblick über die verschiedenen Bereiche der Künstlichen Intelligenz (KI) gegeben werden. In einer Studie des Instituts für Luft- und Raumfahrt (ILR) in Zusammenarbeit mit dem Institut für Produktionsanlagen und Konstruktionstechnik (IPK) über ein wissensbasiertes Diagnosesystem für wiederverwendbare Triebwerke /26/ wurde versucht, eine Übersicht über die Methoden der KI zu geben. Das in Abb. 3-1 gezeigte Ergebnis dieser Übersicht unterscheidet sich von den in der gängigen Literatur zu findenden Einteilungen der KI dadurch, das hier als Kriterium die Methoden der KI und nicht die Anwendungen verwendet wurden. Erst auf unterer Ebene werden auch Beispiele für allgemein übliche Anwendungen der genannten Methoden gemacht. Das AIM Informationssystem fällt in dieser Einteilung unter die regelbasierten Systeme mit Rückwärtsverkettung im Hauptzweig "Symbolische Ansätze".

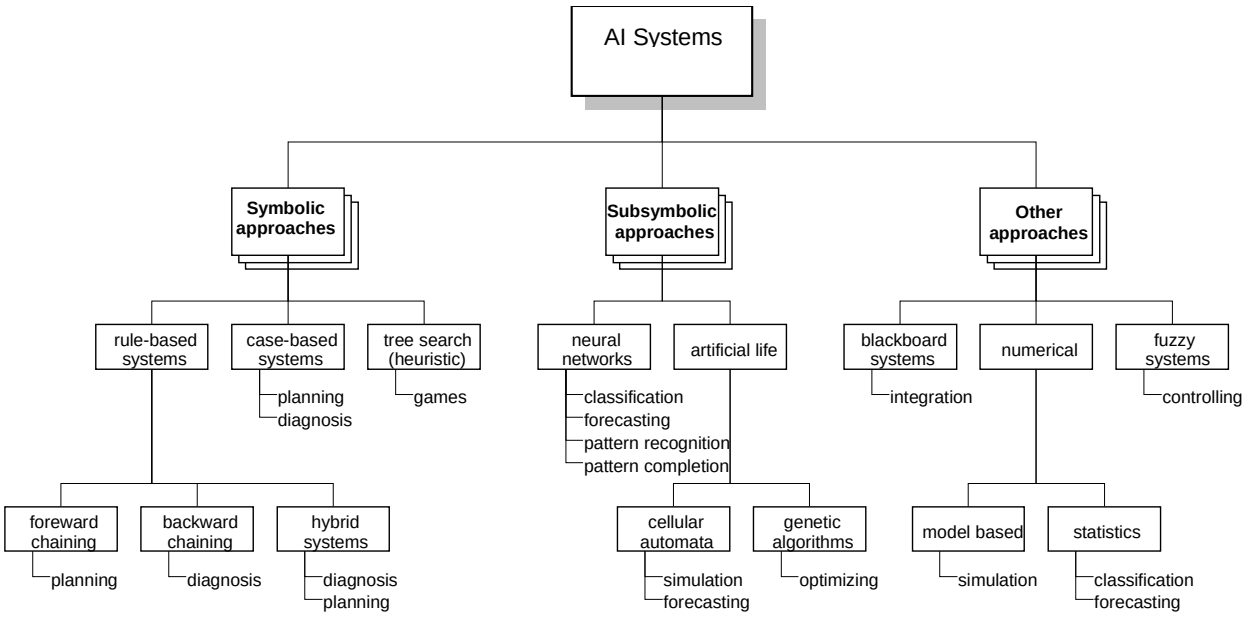


Abb. 3-1 Genealogie der KI Methoden /26/

Wie bereits erwähnt, wird in der Literatur die KI durch ihre Anwendungen definiert. Die folgende Liste von Problemen, bei denen versucht wird mit KI Lösungen zu finden ist nahezu identisch in den Quellen /4/, /48/, /49/ und /52/ zu finden:

- Planning / Planungssysteme
  Die Fähigkeit über die Reihenfolge von Aktionen zu entscheiden, um ein bestimmtes Ziel zu erreichen
- Vision / Computersehen
  Die Fähigkeit Bilder zu verstehen
- Robotics / Robotik
  Die Fähigkeit sich in der realen Welt zu bewegen und zu agieren und eventuell auf veränderte Situationen zu reagieren
- Natural Language / Sprachverarbeitung und Spracherkennung
  Die Fähigkeit mit anderen in einer bestimmten oder beliebigen menschlichen Sprache zu kommunizieren

Im Unterschied zu den bereits genannten Aufgaben mit denen sich die KI beschäftigt, gibt es noch Aufgaben, die ein spezielles Training oder spezielle Fähigkeiten verlangen. Diese Expertenaufgaben sind zum Beispiel:

- Medical or technical diagnosis / Medizinische oder technische Diagnose
- Equipment repair / Reparaturaufgaben
- Computer configuration / Konfiguration technischer Systeme (z.B. Computer)
- Financial planning / Finanzplanung

Die Automatisierung solcher Aufgaben ist vor allem dann nützlich, wenn es an menschlichen Experten mangelt. Solche Lösungen werden als Expertensysteme bezeichnet. In den folgenden Abschnitten wird noch ausführlicher auf diese Anwendung der KI eingegangen.

Wissensrepräsentation

Um die bereits genannten Probleme zu lösen, bedient sich die KI bestimmter Methoden und Techniken. Da einige der Techniken auch im AIM Informationssystem Anwendung finden, soll hier ein Überblick über gängige KI Techniken gegeben werden. Da die Hauptaufgabe von KI Systemen in der Anwendung von Wissen besteht, gibt es verschiedene Techniken der Wissensrepräsentation. Um Probleme innerhalb einer bestimmten Domäne lösen zu können, muss man die Objekte der Domäne kennen sowie Wissen über logische Vorgehensweisen innerhalb der Domäne besitzen. Eine Domäne in der KI ist die Menge von Objekten, welche betrachtet werden, wenn die Bedeutung einer Aussage spezifiziert wird. In /26/ werden verschiedene Repräsentationstechniken in folgende Gruppen zusammengefasst:

- Model based (Modellbasierte Wissensrepräsentation)
  Domänenwissen ist in mathematischen Formeln eingeschlossen.
- Logic based (Logikbasierte Wissensrepräsentation)
  Wissen wird in logischen Formeln wie zum Beispiel „Horn Klauseln“ gespeichert. Logikbasierte Systeme werden in der Regel in PROLOG programmiert.
- Object Oriented (Objektorientierte Wissensrepräsentation)
  Konzepte und Elemente der Domäne werden in Form von Objekten beschrieben. Objekte bestehen aus "Slots" und werden hierarchisch gegliedert. Objektorientierte Systeme stellen Vererbung von deklarativen und funktionalen Eigenschaften der Objekte zur Verfügung.

- Neural Networks (Wissensrepräsentation mit Neuronalen Netzen)
  Ein neuronales Netz simuliert das dynamische Verhalten eines Systems.
- Case-Based (Fallbasierte Wissensrepräsentation)
  Problemlösungen werden aus einer Falldatenbank abgerufen und mit einem neuen Problem verglichen. Elemente von gelösten Problemen werden an die neue Situation angepasst und bilden die Lösung für das neue Problem.

Die für das AIM Informationssystem interessanten Techniken fallen unter die modellbasierten, logikbasierten und objektorientierten Wissensrepräsentationen. Das Expertenwissen ist in einem PROLOG Programm teilweise in Form von mathematischen Formeln (modellbasiert), teilweise als PROLOG Klauseln (logikbasiert) implementiert. Zusätzlich ist in einer relationalen Datenbank Faktenwissen abgespeichert.

Eine sehr detaillierte Zusammenfassung gebräuchlicher Wissensrepräsentationstechniken wird in tabellarischer Form in /52/ genannt. Diese in Tab. 3-1 gezeigte Übersicht ist für die vorliegende Arbeit interessant, da auch auf die Repräsentation von Wissen in der KI durch relationale Datenbanken eingegangen wird. Es wurde bei Beginn der Expertensystementwicklung versucht dieses Faktenwissen in Form von semantischen Netzen und Frames (Rahmen) zu implementieren. Daher wird in den folgenden Absätzen kurz auf diese Wissensrepräsentationstechniken eingegangen. Dies geschieht auch um zu zeigen, warum letztendlich doch nicht auf diese Techniken zurückgegriffen wurde, sondern eine relationale Datenbank Verwendung fand.

Methode	Behandelte Relationen	Inferenzmechanismen	Strenge Organisation?	Hauptsächliche Mängel
Aussagenlogik	Funktionen Bool'scher Wahrheitswerte	Modus Ponens, usw.	Nein	Behandelt nur Beziehungen zwischen Bool'schen Wahrheitswerten und nicht die Aussagen selbst
Konzepthierarchien	"Is-a"	Durchsuchen von Graphen und transitive Erweiterung	Ja	Auf eine Relation beschränkt
Prädikatenlogik	Beliebige Prädikate	Resolution und andere	Nein	Keine Möglichkeit zur Wissensorganisation, schwerfällig bei Informationssteuerung
Rahmen (Frames)	Zwei und dreistellig	Keine Unterstützung	Ja	Nur eine Technik, kein wirkliches Darstellungssystem
Semantische Netze	Zwei und dreistellig	Keine Unterstützung	Nein (mit Ausnahme der Aufteilung)	Kein Standard
Bedingungen	Beliebige Prädikate	Ausbreitung, Erfüllung	Nein	Kein Standard
Produktionsregeln	Wenn-Dann	Regelaktivierung	Nein	Schwerfällig bei nicht-prozeduralem Wissen
Relationale Datenbasen	n-stellig	Selektion, Projektion, Join	Nein	Schwerfällig bei Informationssteuerung

Tab. 3-1 Zusammenfassung und Auswertung von 8 Methoden zur Wissensdarstellung /52/

Semantische Netze

Semantische Netze wurden Anfang der 60er Jahre entwickelt und sind inzwischen eine sehr weit verbreitete Wissensrepräsentationstechnik. In einem semantischen Netz wird das Wissen graphisch in Form einer Art Baum- oder Netzstruktur dargestellt. Die Knoten des Netzes stellen Objekte oder Konzepte dar, die Verbindungen zwischen den Knoten repräsentieren Relationen zwischen den Objekten. Eine spezielle Form der Relation ist die der Unterklasse (subclass). Durch diese Form der Relation wird eine Klassenhierarchie erzeugt, in der die Unterklassen die Eigenschaften der Oberklassen (oder Elternklassen) erben. Eine weitere spezielle Relation ist die Instanziierungsrelation (instance), mit der eine Verbindung zwischen einem Objekt und seiner Elternklasse hergestellt wird. Eigenschaften von Objekten werden mit "besitzt", "Farbe", "besteht aus" Relationen erzeugt. Die Abb. 3-2 zeigt ein Beispiel für ein einfaches semantisches Netz. Dieses Netz repräsentiert das Wissen, das Reptilien und Säugetiere Tiere sind, das Säugetiere Köpfe haben und das ein Elefant ein großes, graues Säugetier ist. Clyde und Nellie sind beide Elefanten und Nellie mag Äpfel.

Semantische Netze erlauben die Wissensrepräsentation von Objekten und ihren Beziehungen untereinander in einer relativ einfachen Weise. Die Darstellung als Graph erlaubt (bei einfachen Systemen) einen guten Überblick. Allerdings sind die ableitbaren Informationen relativ beschränkt. In der Regel können nur Eigenschaften und Vererbungen abgeleitet werden. Außerdem sind semantische Netze keine gute Wahl, wenn es um die Repräsentation von komplexen Strukturen und großen Mengen von Wissen und Fakten geht.

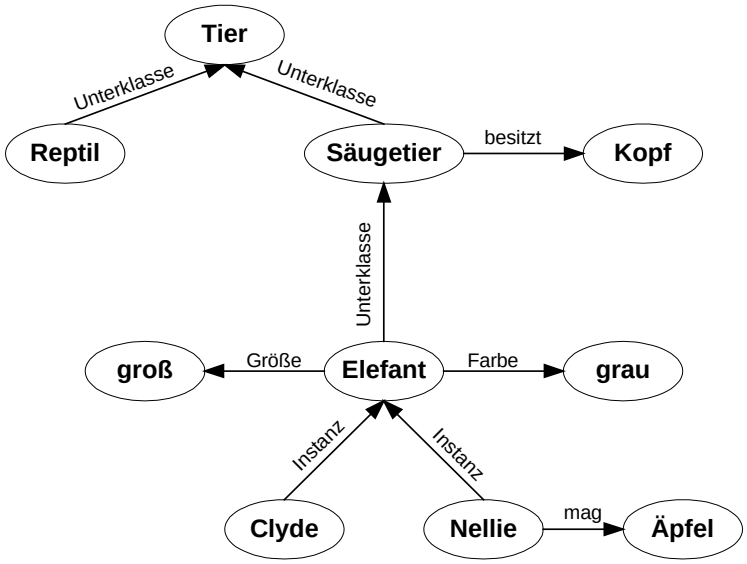


Abb. 3-2 Ein einfaches semantisches Netz /4/

Frames (Rahmen)

Rahmen sind eine Variante der semantischen Netze und werden häufig für die Repräsentation von Fakten in Expertensystemen eingesetzt /4/,/48/,/52/. Sie ähneln stark den von Programmiersprachen bekannten Datentypen (PASCAL: record; C: struct). Da Rahmen in ihrer einfachsten Form Vererbung unterstützen, sind objektorientierte Programmiersprachen und Konzepte ideal geeignet für die Wissensrepräsentation durch Rahmen. In der OOP (objektorientierte Programmierung) werden Objekte mit speziellen Eigenschaften (Properties) und Fähigkeiten (Methoden oder methods) versehen. Die einzelnen Objekte können durch Anordnung in einer aus Klassen und Unterklassen bestehenden Objekthierarchie ihre Eigenschaften und Methoden an andere Objekte/Klassen vererben. Ein Objekt ist dann immer die Instanz einer bestimmten Klasse. In Abb. 3-3 ist ein einfaches Beispiel mit 3 Rahmen gezeigt, welches sich an dem in Abb. 3-2 gezeigten Beispiel aus dem Bereich der semantischen Netze orientiert. Säugetier, Elefant und Nellie sind Objekte in einem Rahmensystem. Säugetier und Elefant sind Klassen, Nellie ist eine Instanz der Klasse Elefant.

Säugetier	
Unterklasse:	Tier
besitzt	Kopf

Elefant	
Unterklasse:	Säugetier
Farbe:	grau
Größe:	groß

Nellie	
Instanz:	Elefant
mag:	Äpfel

Abb. 3-3 Drei einfache Frames /4/

Will man einfache Frames für komplizierte technische Objekte (wie z.B. Objekte aus der Raumfahrt) verwenden, stößt man schnell an Grenzen. Daher werden die einfachen Frames erweitert, um Defaults (Standardwerte; manchmal wird Defaults auch mit Vorbelegung übersetzt /52/) und Multiple Inheritance (Mehrfachvererbung). Bei der Verwendung von Standardwerten vermeidet man die Vererbung aller Eigenschaften der Oberklasse auf die Unterklasse, indem man bestimmte Eigenschaften als Standardwerte kennzeichnet, die von der Unterklasse überschrieben werden können. In Abb. 3-4. sind solche Werte durch "*" gekennzeichnet. Hier wird die Eigenschaft "hat Fell=ja" von Säugetieren bei der Unterklasse Elefant durch "hat Fell=nein" überschrieben. Das Objekt "Clyde" als Instanz der Klasse Elefant ist eine Ausnahme von der typischen "Farbe=grau" Eigenschaft.

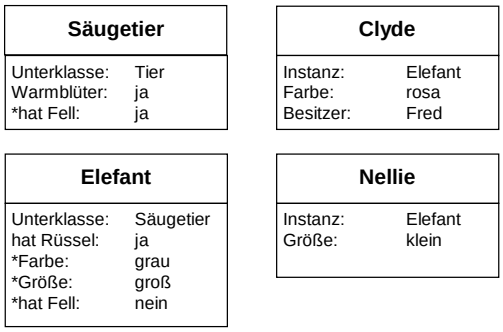


Abb. 3-4 Frames mit Default-Werten /4/

Wenn Mehrfachvererbung eingesetzt wird, kann eine Unterklasse von mehreren Oberklassen Eigenschaften erben. Die Abb. 3-5 zeigt ein System aus drei Rahmen, in dem sowohl Standardwerte als auch Mehrfachvererbung verwendet werden. Das Objekt "Clyde" ist eine Instanz von zwei Klassen und erbt von beiden die jeweiligen Eigenschaften. Wenn man Mehrfachvererbung zulässt, handelt man sich eine Reihe von Komplikationen ein. Zum Beispiel muss festgelegt sein, in welcher Reihenfolge eventuell doppelt vorhandene Eigenschaften überschrieben werden, oder was mit möglicherweise widersprüchlichen Angaben geschehen soll.

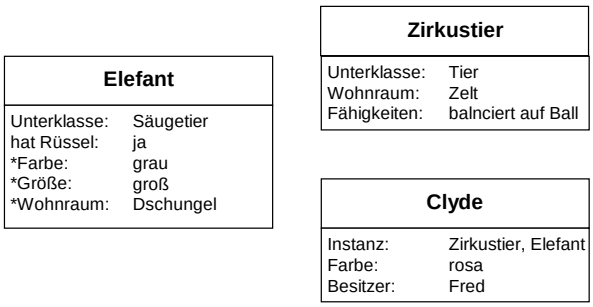


Abb. 3-5 Frames mit Mehrfachvererbung /4/

Relationale Datenbanken

Relationale Datenbasen eignen sich besonders gut zur Bearbeitung umfangreicher, regelmäßig strukturierter Informationsmengen. In einer relationalen Datenbank werden bestimmte Beziehungen abgelegt, aus denen bei Bedarf implizite Zusammenhänge ermittelt werden können. Datenbanksysteme stellen zuverlässige Transformationen zur Verfügung wie z.B. Selektion, Projektion, und Bildung von Joins. Diese werden für Abfragen und in beschränktem Umfang auch für Inferenzen eingesetzt. Wenn die Operationsmethoden des Datenbanksystems mit leistungsfähigeren Inferenzmethoden kombiniert werden (z.B. bei der Resolution mit Prädikatenlogik), erhält man ein gleichermaßen intelligentes wie schnelles wissensbasiertes System /52/. Genau dieses wurde in dieser Arbeit getan, indem das sehr umfangreiche Faktenwissen stark strukturiert in einer relationalen Datenbank gespeichert worden ist und die Inferenz-

methode durch ein PROLOG Programm realisiert wurde, welches über eine Programmierschnittstelle direkt mit der Datenbank kommuniziert.

Probleme

Die ursprünglich geplante Verwendung von Rahmen zur Wissensspeicherung für das Expertensystem scheiterte hauptsächlich am Umfang der Daten. Das verwendete MacProlog kann bei beschränktem Speicherumfang (RAM) nur mit Vorbehalt verwendet werden. Bei der Verarbeitung sehr großer Dateien kommt es außerdem zu starken Leistungseinbussen. Außerdem werden Rahmen von MacProlog nicht von Haus aus unterstützt. Es ist zwar möglich Rahmen zu verwenden, jedoch steigt der Programmieraufwand unverhältnismäßig stark.

Ein weiterer Nachteil von Rahmen ergab sich über Umwege durch die Notwendigkeit nichtlinearer Vererbung. Während der Arbeiten an der Klassifikationskomponente stellte sich sehr schnell heraus, dass Objekte der Raumfahrt nicht in eine lineare Klassenhierarchie unterteilt werden können. Auf diesen Punkt wird in den nachfolgenden Abschnitten noch ausführlich eingegangen. Dadurch wurde eine Verwendung von Standardwerten und Mehrfachvererbung zwingend notwendig. Da bei Verwendung von Rahmen außerdem keine impliziten Ableitungen möglich, sondern nur durch Skripte oder Methoden zu realisieren sind, ergab sich insgesamt ein zu großer Entwicklungsaufwand.

Die Verwendung von semantischen Netzen scheitert ebenfalls an den sehr großen Datenmengen. Auch können die komplexen Relationen zwischen den Objekten im semantischen Netz nicht ausreichend dargestellt werden. Ein weiterer Nachteil der semantischen Netze ist das Fehlen von Vererbungsmöglichkeiten.

Aus den angeführten Gründen und weil für das verwendete MacProlog eine Datenbankschnittstelle rechtzeitig verfügbar war, wird die im Rahmen dieser Dissertation aufgebaute relationale Datenbank AIM verwendet.

Prädikatenlogik und PROLOG

Da für die Expertenregeln PROLOG als Programmiersprache gewählt wurde, soll hier sehr kurz auf die von PROLOG verwendete Prädikatenlogik eingegangen werden. Außerdem werden die beiden unterschiedlichen Ansätze für die Verwendung von Regeln vorgestellt.

PROLOG verwendet Prädikatenlogik erster Ordnung für die Speicherung und Darstellung von Wissen. Es handelt sich um Ausdrücke aus der Prädikatenlogik, sogenannte Hornklauseln. Hornklauseln im zielorientierten Format werden in PROLOG verwendet /52/. Eine Hornklausel ist eine Klausel mit genau /40/ bzw. höchstens /52/ einem nicht-negierten Literal. Beispiele für Hornklauseln sind:

$$\neg A \vee \neg B \vee \neg C \vee X$$

A,B,C,X sind Aussagen

Diese Klauseln lassen sich umformen und als Regel mit einem Implikationspfeil schreiben:

$$A \wedge B \wedge C \rightarrow X$$

In der PROLOG Notation ersetzt ein Komma das "und" und der Implikationspfeil wird umgedreht:

$$X \leftarrow A, B, C.$$

Die Anwendung des Resolutionsverfahrens ist relativ simpel, da zur Herleitung eines Literals nur eine Hornklausel gefunden werden muss, in der das Literal auf der linken Seite steht, und dann die Literale der rechten Seite hergeleitet werden müssen. Der Abarbeitungsmechanismus entspricht der Rückwärtsverkettung von Regeln. Die Hauptarbeit ist die Unifikation, d.h. Substitutionen zu finden, die zwei Literale mit Variablen gleich machen (matching). Falls es mehrere Hornklauseln gibt, deren linke Seiten mit dem aktuellen Ziel unifiziert werden können, probiert PROLOG nacheinander alle Substitutionen aus

und kann so alle Lösungen zu einem Problem finden. Sackgassen werden durch Backtracking überwunden /40/. Dies verhindert allerdings nicht das Hineinlaufen in Endlosschleifen. Die Überwindung dieses Problems war bei der Realisierung des Expertensystems zur Ergänzung des Faktenwissens von entscheidender Bedeutung.

PROLOG ist zur Lösung von Problemen gedacht, die sich mit Objekten und mit Beziehungen zwischen Objekten befassen. Der Programmierer spezifiziert eine Menge von Fakten und Regeln und kann dann Fragen in Form logischer Ausdrücke stellen, die das System beantwortet, indem es versucht Variable durch gewisse Terme zu ersetzen (Unifikation), um Fakten und Bedingungsteil von Regeln mit dem Zielausdruck zur Deckung zu bringen (matching); dann führt das System Resolutionen durch, es ersetzt den Zielausdruck durch ein "gematchtes" Fakt bzw. den Ausführungsteil der "gematchten Regel" /49/.

Regelbasierte Systeme

Regelbasierte Systeme repräsentieren Wissen nicht in einer statischen, deklarativen Art sondern als eine Sammlung von Anweisungen, die angeben, was in einer bestimmten Situation zu tun ist oder welche Schlüsse man in verschiedenen Situationen ziehen kann. Ein regelbasiertes System besteht aus einer Sammlung von Wenn-Dann Regeln, einer Anzahl Fakten und einem "Interpreter", der die Anwendung der Regeln kontrolliert. Im Gegensatz zum Gebrauch von Wenn-Dann Regeln in konventionellen Programmiersprachen, wo diese Teil einer sequentiellen Verarbeitung mit Verzweigungen sind, wird in KI Systemen jede Regel als ein eigenständiges Stück Wissen verstanden, welches bei Bedarf benutzt werden kann (s.a. Abb. 3-6).

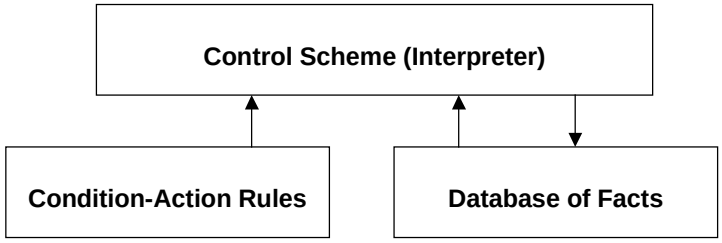


Abb. 3-6 Architektur regelbasierter Systeme /4/

Es gibt zwei wichtige Arten von Interpretern: Interpreter mit Vorwärtsverkettung oder Interpreter mit Rückwärtsverkettung. In vorwärts verkettenden Systemen wird mit einigen Anfangsfakten gestartet und die Regeln werden benutzt um Schlüsse zu ziehen, welche auf den gegebenen Fakten beruhen. In rückwärts verkettenden Systemen startet man mit einer Hypothese oder einem Ziel und versucht Regeln zu finden, die die Richtigkeit dieses Ziels beweisen. Die Vorwärtsverkettung wird auch datengesteuertes Verfahren oder Bottom-up Verfahren genannt. Die Rückwärtsverkettung bezeichnet man auch als zielgesteuertes oder Top-Down Verfahren /13/.

Welche von beiden Techniken man auswählt, hängt von den Eigenschaften der Regelbasis, von den Anfangsfakten und davon wie viele mögliche Hypothesen betrachtet werden ab. In dem in dieser Arbeit betrachteten Fall würde sich eigentlich die Vorwärtsverkettung anbieten, da die Anfangsfakten sehr umfangreich sein können. Da jedoch PROLOG auf Rückwärtsverkettung mit Backtracking basiert, musste bei der Programmierung versucht werden PROLOG eine Art Vorwärtsverkettung aufzuzwingen. Für die eigentliche Problemlösung benutzt PROLOG natürlich seine eigenen Problemlösungsstrategien.

3.2 Expertensysteme

Expertensysteme sind Programme, mit denen das Spezialwissen und die Schlussfolgerungsfähigkeit qualifizierter Fachleute auf eng begrenzten Aufgabengebieten nachgebildet werden soll. Expertensysteme benötigen detaillierte Einzelkenntnisse über das Aufgabengebiet und Strategien, wie dieses Wissen zur Problemlösung benutzt werden soll. Die Trennung zwischen Wissen und Problemlösungsstrategie ist charakteristisch für die Architektur von Expertensystemen und wissensbasierten Systemen. /40/.

Wie Abb. 3-7 zeigt unterscheidet man bei wissensbasierten Systemen zwischen Wissensbasis und Problemlösungskomponente (Inferenzmaschine). Die strikte, auch programmtechnische, Trennung zwischen Wissen und einer speziellen Inferenzmaschine ermöglicht es grundsätzlich auch Erklärungen über das Wissen zu erzeugen, z.B. darüber, wie bestimmte Schlussfolgerungen erreicht wurden.

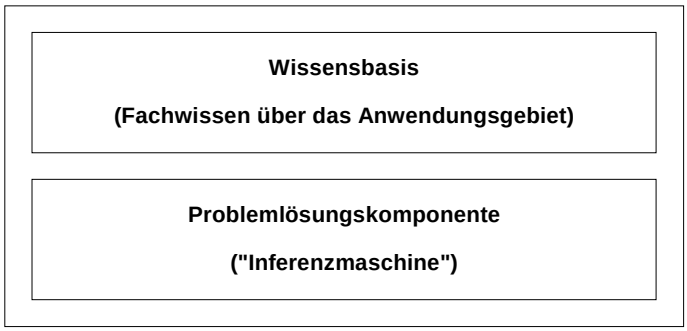


Abb. 3-7 Kern eines wissensbasierten Systems /22/

Expertensysteme sind Hilfsmittel für Experten. Ein Expertensystem muss auf Aufforderung erklären können, wie es zu seiner Antwort kam. Und dies in einer Sprache, die der Experte versteht, damit er das System gegebenenfalls überprüfen kann /46/. Eine Einordnung in den Bereich der KI Programme zeigt Abb. 3-8. Die Abgrenzung zwischen wissensbasierten Systemen und Expertensystemen ist nicht immer so einfach wie es die Grafik suggeriert. Auch bei dem später vorgestellten Expertensystem ist der einzige Unterschied zum wissensbasierten System, das es das anwendungspezifische Wissen nicht nur bereitstellt, sondern auch für die Problemlösung anwendet.

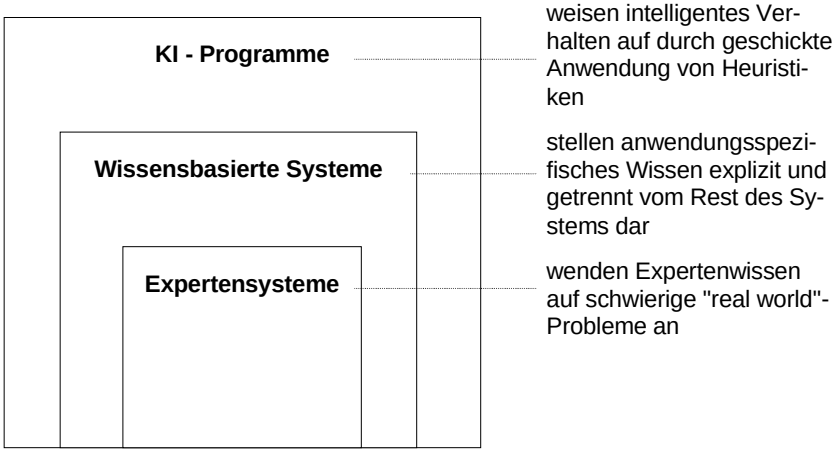


Abb. 3-8 Einordnung von Expertensystemen nach Waterman /22/

Die Brückenfunktion der wissensbasierten Systeme und die unscharfen Grenzen zwischen konventioneller Programmierung und KI zeigt Abb. 3-9. Auch hier sei darauf hingewiesen, dass das AIM Informationssystem alle dargestellten Bereiche abdeckt. Der Datenbankteil mit seinen Abfragewerkzeugen stellt ein typisches "Klassisches Informationssystem" dar. Durch die Integration des Regelwissens in die Datenbank kann man auch von einem "Wissensbasierten System" sprechen. Allerdings gibt es keine

kontextspezifische Bereitstellung von Regelwissen für den reinen Datenbankbereich. Dies kann allenfalls vom "Expertensystemteil" behauptet werden, der mit seiner Erklärungskomponente dem Benutzer kontextspezifisch (auf die jeweilige Datenbankabfrage bezogen) Teile seines Regelwissens bereitstellt und erläutert.

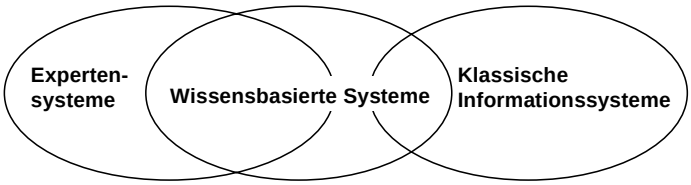


Abb. 3-9 Spektrum wissensbasierter Systeme /22/

Bei der Untersuchung von Merkmalen aktueller Expertensysteme (Stand 1992) wird in /22/ festgestellt:

- Die Erklärungsfähigkeit ist zwar häufig vorhanden, aber von mäßiger Qualität.
- Lernfähigkeit besitzen die Systeme nach dem heutigen Stand der Kunst nicht oder nur äußerst rudimentär.
- Reorganisation des Wissens ist bisher in der Regel nicht möglich; insbesondere sind die Systeme nicht in der Lage, inkonsistentes Wissen zu behandeln oder sich über Regeln hinwegzusetzen.
- Die Relevanz seines Wissens für eine bestimmte Aufgabe kann ein Expertensystem nicht bestimmen. Innerhalb seiner Domäne kann ein Expertensystem zwar unter Umständen komplizierteste Probleme besser als ein Mensch lösen; es kann aber nicht feststellen, ob es für ein gegebenes Problem überhaupt kompetent ist.
- Expertensysteme verlassen ihr Wissensgebiet nicht "mit Anstand", im Gegenteil: Beim Überschreiten der Grenzen fällt die Leistungsfähigkeit ganz überstürzt ab.

Aufbau von Expertensystemen

Alle Quellen der Fachliteratur sind sich über den grundsätzlichen Aufbau von Expertensystemen einig. Einige Quellen unterteilen in Programmmodule andere sprechen von Funktionsbereichen. Bei den in Abb. 3-10 gezeigten Komponenten handelt es sich nicht um Programme oder Programmmodule sondern um Funktionsbereiche. Im folgenden werden die einzelnen Funktionsbereiche etwas genauer vorgestellt.

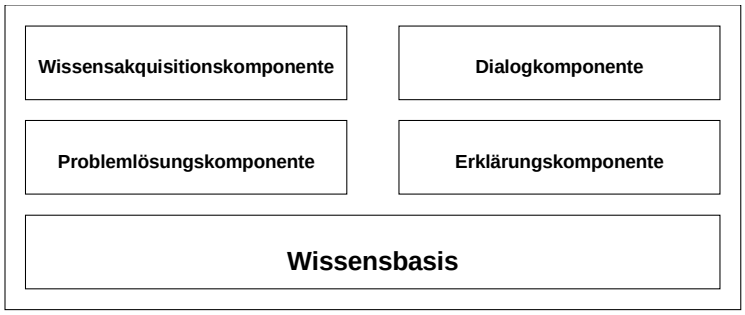


Abb. 3-10 Aufbau eines Expertensystems /22/

Wissensakquisitionskomponente

Mit der Wissensakquisitionskomponente oder Wissenserwerbskomponente, wird das Wissen in die Wissensbasis eingebracht. Die Wissensakquisitionskomponente dient der allgemeinen Manipulation der Wissensbasis. Es sollte neues Wissen aufgenommen werden können, vorhandenes Wissen sollte geändert werden können und nicht mehr aktuelles oder fehlerhaftes Wissen sollte entfernt werden können. Laut

/22/ verfügen die meisten Expertensysteme jedoch nicht über eine eigene Wissensakquisitionskomponente. Statt dessen werden speziell angepasste Editoren, die auf die jeweilige Art der Wissensdarstellung zugeschnitten sind verwendet. Manchmal werden auch zusätzliche Werkzeuge für die Wissensakquisition verwendet.

Von den in Abb. 3-11 gezeigten Grundmodellen der Wissensakquisition werden in der Regel nur die ersten beiden Arten verwendet. Die automatische Wissensakquisition fällt in den Bereich "Maschinelles Lernen" und ist Gegenstand verschiedener Forschungsprojekte. Die mittlere Variante mit einer intelligenten Akquisitionskomponente ist eigentlich der wünschenswerte Fall. In der Regel jedoch wird die Wissensbasis von einem Wissensingenieur (knowledge engineer) erstellt und gepflegt, der das Fachwissen durch verschiedene Techniken (Fragebogen, Interview, Literatur, etc.) erlangt und für die Verwendung durch das Expertensystem aufarbeitet. Die Abbildung müsste eigentlich noch durch den nicht seltenen Fall ergänzt werden, in dem Experte und knowledge engineer identisch sind.

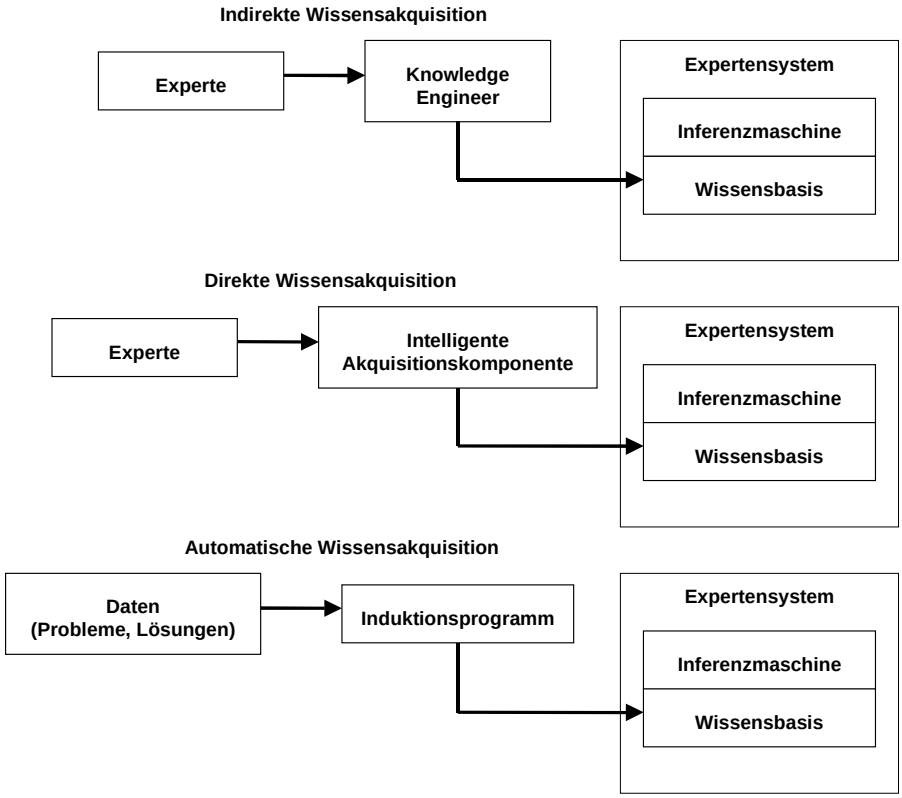


Abb. 3-11 Grundmodelle der Wissensakquisition /22/

Wissensbasis

Die Wissensbasis enthält das Fachwissen des Experten über das Anwendungsgebiet (domain specific knowledge) /22/. Auf die verschiedenen Formen der Darstellung von Wissen wurde bereits im vorhergehenden Abschnitt ausführlich eingegangen.

In Abb. 3-12 sind noch einmal die wichtigsten Techniken zur Wissensdarstellung unter Berücksichtigung der beiden Aspekte, deklarative und prozedurale Wissensrepräsentation gegenübergestellt. Das AIM Expertensystem benutzt zur Wissensdarstellung Objekt-Attribut-Wert-Tripel (O-A-W-Tripel) und Prädikatenlogik. Die Einträge der Datenbank werden nach der SQL-Abfrage durch das PROLOG Programm als O-A-W-Tripel behandelt, und (bei Benutzung der Cache Funktion) als PROLOG Fakten in die Regelbasis aufgenommen. Alle Produktionsregeln sind als Hornklauseln in Prädikatenlogik im PROLOG Programm implementiert.

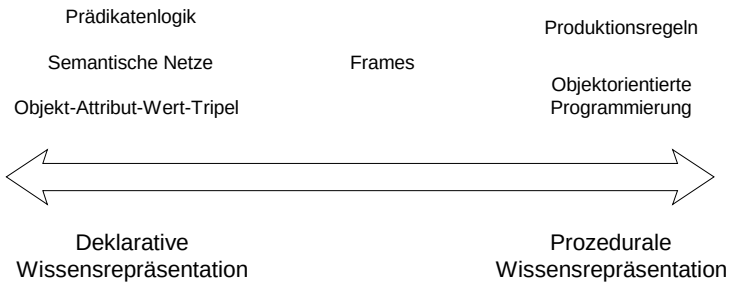


Abb. 3-12 Deklarative und prozedurale Wissensrepräsentation /22/

Problemlösungskomponente

Die Problemlösungskomponente benutzt das in der Wissensbasis enthaltene Wissen um Schlussfolgerungen zu ziehen und neue Wissenseinheiten zu erzeugen, die gegebenenfalls der Wissensbasis hinzugefügt werden. Die Reihenfolge in der das Wissen ausgewertet wird, ermittelt die Problemlösungskomponente auf der Grundlage einer bestimmten Abarbeitungsstrategie. In AIM Expertensystem wird als Problemlösungskomponente PROLOG verwendet. Dies bestimmt auch gleichzeitig die Abarbeitungsstrategie.

Dialogkomponente

Die Dialogkomponente stellt die Schnittstelle zwischen Benutzer und Expertensystem dar. Sie steuert den Dialog zwischen dem Benutzer und dem System. Die Hauptaufgabe dieser Komponente besteht in der benutzerfreundlichen Aufbereitung und Darstellung des Wissens. Zum Teil werden heute auch bei Expertensystemen, die von den kommerziellen Anwendungsprogrammen gewohnten GUI (Graphical User Interface) benutzt, welche komfortable Benutzeroberflächen mit grafischen oder tabellarischen Elementen verwenden. In der Zukunft sind auch natürlichsprachliche Dialogschnittstellen denkbar. Im Expertensystemteil des AIM Informationssystems wird HyperCard als lokales GUI und HTML als Netzwerk GUI verwendet. Man kann über das WWW auf das Informationssystem zugreifen und unter Verwendung des entsprechenden Browsers erhält man alle Ergebnisse grafisch dargestellt.

Erklärungskomponente

Die Erklärungskomponente gibt auf Anforderung Begründungen ab, wie die Ergebnisse oder Zwischenergebnisse abgeleitet wurden. Das Vorhandensein einer leistungsfähigen Erklärungskomponente ist besonders wichtig für die Akzeptanz des Expertensystems. Der Endbenutzer akzeptiert die Lösungen, die ihm das System liefert, eher, wenn er sie auch nachvollziehen kann. Die Erklärungskomponente soll ihn deshalb in die Lage versetzen, zu überprüfen ob einzelne Schlussfolgerungen und das Endergebnis plausibel sind. Häufig ist der Endbenutzer der Experte, und auch bereits bei der Erstellung des Systems kommt der Experte ständig mit dem System in Berührung; mit Hilfe der Erklärungskomponente kann der Experte dann zum Beispiel feststellen, ob seine Schlussweisen vom System korrekt nachgebildet werden.

Die Erklärung sieht häufig so aus, dass das System beispielsweise angibt, welche Wissenseinheiten in welcher Reihenfolge für die Schlussfolgerung herangezogen wurden (z.B. welche Regeln nacheinander abgearbeitet wurden) /22/. Im AIM System werden die verwendeten Fakten aus der Datenbank, die angewandten Regeln in der Reihenfolge ihrer Anwendung, sowie die von den Regeln gelieferten Zwischenergebnisse übersichtlich dargestellt. Zu jeder einzelnen Regel sind dann noch zusätzliche Informationen abrufbar.

4 Die AIM Datenbank

Im folgenden Teil wird das AIM (AIM = Aerospace Information and Modeling) Informations- und Expertensystem beschrieben. Es werden Lösungen für die im allgemeinen Teil angesprochenen Probleme vorgestellt. Verschiedene Lösungsmethoden werden verglichen und die Auswahl der letztendlich benutzten Methode wird begründet. Die Wissensbasis für das Faktenwissen des Informationssystems wird vorgestellt. In den sich anschließenden Kapiteln werden die Expertensystemkomponenten des AIM Informationssystems beschrieben.

4.1 Arten von Informationen und ihre Speicherung im AIM System

Beim Aufbau der AIM Datenbank kam es zunächst darauf an, festzustellen, welche Arten von Wissen gespeichert werden sollten. Dazu wurde innerhalb einer Bestandsaufnahme festgestellt in welcher Form Wissen am Lehrstuhl vorhanden war. Die folgende Aufzählung listet die verschiedenen Informationsarten, die in einem Informationssystem/Expertensystem vorhanden sein können, auf. Man kann davon ausgehen, dass nahezu alle in der Liste vorhandenen Arten am Institut vorhanden sind.

- Daten (Zahlenwerte)
- kurze/lange Texte
- Bilder/Grafiken
- Filme/Sounds
- Formeln/Gleichungen
- Relationen
- Simulationen/Modelle
- Tabellen
- Schätzungen
- Heuristiken
- Regeln
- Software (Programme)

Ein Teil der Informationen war bereits in strukturierter Form vorhanden. Zum Beispiel waren die Texte durch Inhaltsverzeichnisse und Kapitel vorstrukturiert. Ein Großteil der als Zahlenwerte vorhandenen Daten war in einem datenbankähnlichen Programm (HyperCard für Apple Macintosh) gespeichert. Zusätzliches Material war größtenteils in Tabellen oder zumindest geordneten Listen verfügbar.

Als nächstes wurde eine grobe Einteilung vorgenommen, welche Informationen in der Datenbank gespeichert und welche im Expertensystemteil implementiert werden sollten. Hierbei wurde unterschieden zwischen reinem Faktenwissen und eher regelorientiertem Expertenwissen. Tab. 4-1 zeigt die grobe Verteilung der verschiedenen Arten von Wissen bzw. Informationen auf die Komponenten des AIM Informationssystems.

Fakten	Multimedia	Expertenwissen
Daten (Zahlenwerte) Tabellen Simulationen (Ergebnisse) Zitate	kurze/lange Texte Bilder Grafiken Filme/Sounds	Formeln/Gleichungen Relationen Schätzungen Heuristiken Regeln Software (Programme) Simulationen/Modelle

Tab. 4-1 Grobe Einteilung von Informationen und Wissen

Fakten und Multimedia sind Bestandteil des Informationssystems respektive der Datenbank. Die Multimediadaten sind extern gespeichert, unterliegen also nicht der Kontrolle des RDBMS, sondern befinden sich als Dateien (hauptsächlich Bilder und Texte, aber auch einige Filme) auf einem Rechner, der als WWW Server fungiert. Die Zuordnung der Multimediaobjekte zu den Objekten der Raumfahrt wird dagegen durch Relationen nahezu aller Objekte mit den Tabellen `Picture` und `TextFile` hergestellt. (s.a. Beschreibung der Tabellen `Picture` und `Textfile` in den folgenden Abschnitten).

Auswahl der Hard- und Software

Bei der Wahl der Hard und Software waren die Wahlmöglichkeiten relativ stark eingeschränkt. Zum Einen ließ das Budget des Instituts keine größeren Spielräume bei der Anschaffung neuer Hard- und

Software zu, so dass zwangsläufig auf die bestehende Infrastruktur zurückgegriffen werden musste. Zum Anderen waren die potentiellen Benutzer bereits an bestimmte Software gewöhnt, so dass auch hier eine komplette Umstellung nicht ratsam erschien. Als Hardwareplattform wurde deshalb ein Apple Macintosh Computer (Apple Macintosh II ci) gewählt, da zum Zeitpunkt der Inbetriebnahme des Datenbankservers (bis auf zwei IBM PC-Kompatible Rechner) ausschließlich Apple Macintosh Computer im Fachgebiet verwendet wurden.

Durch die Wahl der Hardware ergaben sich natürlich automatisch Beschränkungen bei der Software. Als Datenbanksoftware wurde der ORACLE Server 6.0 für Macintosh gewählt, da eine Einzelplatzversion bereits am Institut vorhanden war und ein Update auf die leistungsfähigere Serverversion von allen Optionen die Kostengünstigste war. Die Entscheidung, die bisher am Institut verwendete Software HyperCard abzulösen hing auch mit den Nachteilen des bisher verwendeten datenbankähnlichen Programms zusammen, welche in der folgenden Aufzählung zusammengefasst sind:

- keine Mehrbenutzerunterstützung
- nicht Server-basiert (d.h. keine zentrale Datenhaltung)
- Leistungsprobleme bei großen Datenmengen
- keine relationalen Datenstrukturen

Die Vorteile des relationalen Datenbankservers aus dem Hause ORACLE gegenüber der weniger leistungsfähigen HyperCard Software liegen auf der Hand:

- Relationale Datenbank
- Mehrbenutzerunterstützung
- Client/Server Architektur mit Netzwerkunterstützung
- Multiprotokolltechnik (AppleTalk, Ethertalk, TCP/IP)
- Unterstützung aller gängigen Betriebssysteme (Unix, MacOS, Windows, etc.)
- Client Software für alle Betriebssysteme verfügbar

Für den Zugriff auf die Informationen und die Datenmanipulation wurde HyperCard gewählt. Trotz der o.a. Nachteile von HyperCard als "Datenbank" ist es für eine schnelle Softwareentwicklung (rapid prototyping, bzw. rapid application development) sehr gut geeignet. Mit HyperCard lassen sich in kurzer Zeit Programme mit grafischer Benutzeroberfläche entwickeln. Die mangelnde Performance, bedingt durch den fehlenden Compiler, wird durch hervorragende Debugging-Fähigkeiten (bei Interpretersprachen schon immer üblich) wieder wettgemacht. Nicht ohne Grund ist bei der ORACLE Server Software (V6.0.36.1.0) eine HyperCard Schnittstelle unter der Bezeichnung ORACLE Access integriert. Die Datenmanipulationssoftware und die Abfrageprogramme wurden daher in HyperCard entwickelt. Erst später wurde ein WWW Interface (ebenfalls in HyperCard) hinzugefügt, welches einfache Datenbankabfragen über das Internet ermöglicht. Zusätzlich zu der selbst entwickelten Software ist es möglich kommerzielle Programme für die Datenbankabfrage zu benutzen. Voraussetzung hierzu ist die Fähigkeit der verwendeten Software eine der gängigen Schnittstellen zu verwenden (z.B. ODBC = Open Database Connectivity).

4.2 Datenmodell und Struktur der Datenbank

Für den Aufbau der AIM Datenbank hätte sich ein objektorientiertes Datenschema angeboten, da es in der Raumfahrt eine Vielzahl von technischen Systemen gibt, deren Eigenschaften zum Teil stark unterschiedlich sind. Wollte man diese Eigenschaften in einer streng tabellenorientierten Datenbank speichern, stößt man sehr schnell auf erhebliche Probleme. Daher wurde zu Beginn bei der Planung der AIM Datenbank eine objektorientierte Datenstruktur entworfen. Allerdings lassen sich die Objekte der Raumfahrt nicht so einfach in Klassen einteilen wie man meinen könnte (auf das Problem der Klassifizierung von Raumfahrtobjekten wird in Kapitel 1 nochmals detailliert eingegangen). Die Struktur der

AIM Datenbank geht daher auf diesen objektorientierten Entwurf zurück, ist aber auf Grund der Implementierung in einem relationalen Datenbanksystem nur in Ansätzen objektorientiert. Der objektorientierte Entwurf wurde gemacht, als noch nicht feststand, dass ein relationales Datenbanksystem verwendet werden würde. Mit der Verwendung der relationalen Datenbank ORACLE sind natürlich alle Vor- und Nachteile verbunden, die im Kapitel 1 der Arbeit bereits genannt wurden.

Diskurswelt und Datenbasis

Die Diskurswelt des AIM Informationssystems sind alle Objekte der realen Welt, die mit der weltweit betriebenen Raumfahrt in engerer Verbindung stehen. Diese werden durch die an der TU Berlin in den Fachgebieten Raumfahrzeugtechnik (Prof. Lo), Raumfahrtgeräte und -anlagen (Prof. Renner) und Raumflugmechanik (Prof. Priebs) angebotenen Vorlesungen abgedeckt. Diese Objekte umfassen:

- Raumtransportsysteme aller Art (ELVs, RLVs, OTVs)
- Antriebe für alle Arten von Raumfahrtobjekten (Raketen- und Satellitenantriebe)
- Raumfahrtgeräte und –anlagen (Satelliten, Raumstationen)
- Objekte der Raumflugmechanik (Umlaufbahnen, Aufstiegsbahnen)

Die Objekte der realen Welt sollten in ihren in der Raumfahrt anerkannten Klassen in die Datenbank integriert werden. Bei der Umsetzung der Diskurswelt in die Datenbasis sollte darauf geachtet werden, dass die späteren Benutzer die ihnen aus der Diskurswelt bekannten Einteilungen und Bezeichnungen wiedererkennen und somit ein intuitives Verständnis zumindest gefördert wird.

Datenmodell

Neben der bereits genannten Forderung des intuitiven Wiedererkennen der Datenstruktur, sind noch weitere Voraussetzungen zu erfüllen, die Einfluss auf das Datenmodell haben. Da ist zunächst die Forderung der einfachen Erweiterbarkeit. Es sollte möglich sein, alle Objekte der Diskurswelt im Datenmodell abzubilden. Aber auch bereits bekannte, noch nicht abgebildete Objekte sind zu jedem Zeitpunkt in einfacher Form zu integrieren, ohne eine Neu- oder Umstrukturierung vornehmen zu müssen. Auch sollte es möglich sein, zum gegenwärtigen Zeitpunkt noch unbekannte Objekte zu berücksichtigen, da die Raumfahrt eine sehr dynamische technische Fachrichtung darstellt, in der nahezu alles technisch Mögliche und manchmal auch Unmögliche irgendwann als reales System, wenn nicht gebaut so doch intensiv studiert wird. Damit verbunden ist natürlich die Forderung, nicht nur reale Systeme sondern auch Studien zu erfassen.

Obwohl die Eigenschaften von Raumfahrtsystemen extrem vielfältig sind, sollte es möglich sein, *alle* möglichen Informationen flexibel zu speichern. Das heißt eventuell neu hinzukommende Eigenschaften sollen gespeichert werden können, ohne große Veränderungen an der Datenstruktur vorzunehmen. Zusätzlich soll zu jedem einzelnen Zahlenwert die genaue Quellenangabe möglich sein. Durch die Zuordnung der Quelle zu jedem einzelnen Datum ist es möglich, die Herkunft der Daten zurückzuverfolgen. Dies ist notwendig, da man in der Raumfahrt in der Regel mehrere zum Teil widersprüchliche Angaben findet. Das resultiert zum einen aus der allgemein üblichen Geheimhaltung, da es sich auch meist um militärische Projekte handelt, zum Anderen aber auch daraus, dass bis zur Einführung eines technischen Systems in der Raumfahrt, sehr viele Modifikationen und Alternativen untersucht werden, die dann in der Literatur mit dem fertigen System verwechselt werden und sich natürlich in den Eigenschaften unterscheiden. Dann ist es auch üblich, bereits eingeführte Systeme kontinuierlich zu verbessern und in Details Veränderungen vorzunehmen, ohne dass eine Versionsänderung von der Öffentlichkeit unbedingt zur Kenntnis genommen wird.

Datenstruktur

Entity-Tabellen (E-Tabellen)

Um die genannten Anforderungen zu erfüllen wurde ein Entity-Relationship Datenmodell gewählt. Das Entity-Relationship Modell wird den Anforderungen gerecht und ist das am meisten verwendete bei relationalen Datenbanken. Die Entity-Tabellen stellen die Hauptstützen des Modells dar. Sie enthalten die Objektklassen oder Objekttypen der Diskurswelt mit allen Eigenschaften, die keine Zahlenwerte und allen Objekten der Klasse gemeinsam sind. Zum Beispiel steht das reale Objekt "Ariane 44LP" der Diskurswelt in der AIM Datenbank als Eintrag in der Tabelle `Launcher`. Die Tabelle `Launcher` ist die Zusammenfassung aller Entities vom Objekttyp `Launcher`. Eine Übersicht über alle Entity-Tabellen der AIM Datenbank zeigt Tab. 4-4.

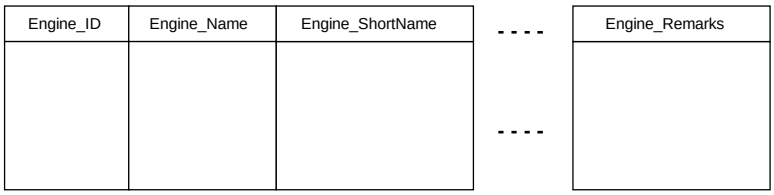


Abb. 4-1 Typischer Aufbau einer Entity-Tabelle der AIM Datenbank am Beispiel `Launcher`

Die Entity-Tabellen enthalten immer die Spalten *`Objekttyp`*`_ID` (mit Objekttyp = `Launcher`, `Stage`, `Engine`, etc.) als primären Schlüssel, mit dem `Constraint NOT NULL` um Einträge in dieser Spalte zu erzwingen und einem `UNIQUE INDEX` namens `Ix`*`Objekttyp`* auf eben dieser Spalte um doppelte `IDs` zu vermeiden. Außerdem existiert immer eine Spalte *`Objekttyp`*`_Name`, welche ebenfalls mit dem `Constraint NOT NULL` und einem `UNIQUE INDEX` namens `Ix`*`Objekttyp`*`_Name` versehen ist, um doppelte Namen zu verhindern. Einige Entity-Tabellen (`Attribute`, `Engine`, `Fairing`, `Launchsite`, `Nation`, `Orbit`, `Organization`, `Substance`) enthalten zusätzlich noch eine Spalte namens *`Objekttyp`*`_ShortName` mit den Abkürzungen oder speziellen Bezeichnungen des jeweiligen Objekts.

Datentabellen

Die Datentabellen stellen zwar durch das Vorhandensein des Fremdschlüssels `Reference_ID` eine Relation zwischen `Reference` und dem jeweiligen Objekttyp her, dies war aber nicht die ursprüngliche Absicht. Eigentlich sind die Datentabellen dafür gedacht einen Zusammenhang zwischen komplexen Attributen eines Objekttyps herzustellen und die Attributwerte mit dem Objekttyp `Reference` zu verbinden. Nach den in Kapitel 2.2.2 erwähnten Arten von Relationen sind die Datentabellen eine Mischung aus Fremdschlüssel-Beziehungen und Erweiterungen der Entity-Tabellen durch Zusammenhänge zwischen komplexen Attributen. Sie enthalten zwar außer dem Fremdschlüssel der Entity-Tabelle für die sie eine Erweiterung darstellen auch den Fremdschlüssel `Reference_ID`, aber die eigentlichen Daten (Eigenschaftswerte) sind in gesonderten Tabellen gespeichert, um

- beliebige Eigenschaften mit beliebigen Objekten zu verknüpfen und
- jedem Zahlenwert bzw. jedem Eintrag eine Quelle zuordnen zu können.

Dabei sind die Eigenschaften nochmals gruppiert. Ursprünglich wurde die Einteilung nach den Einheiten der jeweiligen Eigenschaften und nach der Art der Werte vorgenommen. Daraus ergab sich folgende Einteilung (x steht für den Objekttyp, z.B. `Launcher`):

- Kostendaten (Tabelle: *`x`*`_Cost`, Wert: Realzahl, Einheit: "MY" Mannjahre)
- Datumswerte (*`x`*`_Date`, Datum ,keine Einheit)
- Geometriedaten (*`x`*`_Geometry`, Realzahl, Einheit: "m")
- Massendaten (*`x`*`_Mass`, Realzahl, Einheit: "kg")
- Leistung (*`x`*`_Performance`, Realzahl, Einheit: alle außer den Erstgenannten)

Durch diese Einteilung erübrigte sich eine Einheitenspalte bei den Tabellen *Objekttyp*_Mass und *Objekttyp*_Geometry, da "m" und "kg" als Einheit vorausgesetzt wurden. Es stellte sich jedoch heraus, dass die meisten Ingenieure Volumen und Flächen unter Geometrie einordnen und nicht unter Leistung, so dass die Tabelle Geometry eine Einheitenspalte (*Objekttyp*_GeomUnit) erhielt, in der "m", "m^2"oder "m^3" eingetragen wird.

Jede dieser Tabellen enthält eine Spalte für die Reference_ID, um eine Relation zur Entity-Tabelle Reference herzustellen. In Reference sind die Quellen für die Zahlenwerte als Literaturhinweise gespeichert. Die Tabelle Reference wird auch für Suchfunktionen in der Bibliothek des Fachgebiets benutzt.

Engine_Cost

Engine_ID	Reference_ID	Engine_CostType	Engine_CostValue	Engine_CostUnit	Engine_CostDefault	Engine_CostRemarks
718	3245	Development Cost	5	MY	y	
718	234	Manufacturing Cost	450000	US$1985	y	
718	124	Development Cost	4,5	MY	n	
145	43	Refurbishment Cost	0,5	ECU1990	y	

Engine_Date

Engine_ID	Reference_ID	Engine_DateType	Engine_DateValue	Engine_DateDefault	Engine_DateRemarks
213	13	First Flight Test	12-JAN-76	y	
12	233	Last Flight	23-MAR-80	y	
34	654	Development Start	16-NOV-87	y	
754	54	Production Start	5-OCT-66	y	

Engine_Geometry

Engine_ID	Reference_ID	Engine_GeomType	Engine_GeomValue	Engine_GeomUnit	Engine_GeomDefault	Engine_GeomRemarks
718	3245	Chamber Diameter	1,2	m	y	
718	234	Nozzle Length	2,3	m	y	
718	124	Engine Height	2,2	m	n	
145	43	Nozzle Exit Area	1,5	m^2	y	

Engine_Mass

Engine_ID	Reference_ID	Engine_MassType	Engine_MassValue	Engine_MassDefault	Engine_MassRemarks
718	3245	Chamber Mass	200	y	
718	234	Dry Mass	500	y	
718	124	Total Mass	1200	n	
145	43	Nozzle Mass	60	y	

Engine_Performance

Engine_ID	Reference_ID	Engine_PerfType	Engine_PerfValue	Engine_PerfUnit	Engine_PerfDefault	Engine_PerfRemarks
718	3245	Mixture Ratio	5	kgO/kgF	y	
718	234	Specific Impulse	320	s	y	
718	124	Thrust Vacuum	2000	kN	n	
145	43	Mass Flow Rate	200	kg/s	y	

Abb. 4-2 Aufbau der Datentabellen Cost, Date, Geometry, Mass, Performance der AIM Datenbank am Beispiel Engine

Relationship (R-Tabellen) und Entity-Relationship (ER-Tabellen) Tabellen

Die Verknüpfungen zwischen den Objekttypen werden je nach Komplexität der Relation modelliert (s.a. Kapitel 2.2.2). Mit reinen Relationship-Tabellen (das sind Tabellen die nur Fremdschlüssel enthalten) können alle Komplexitäten des einfachen ER-Modells gebildet werden. Tab. 4-2 zeigt die Zusammenhänge für zweispaltige R-Tabellen.

Komplexität	Modellierung
1:1	Ein UNIQUE INDEX ist definiert der beide Fremdschlüssel umfasst
1:n, n:1	Ein UNIQUE INDEX ist definiert der einen der beiden Fremdschlüssel umfasst
n:m	Kein UNIQUE INDEX ist definiert
Beide Spalten der R-Tabelle sind immer mit dem Constraint NOT NULL definiert, dürfen also keine leeren Zeilen (Nullwerte) enthalten	

Tab. 4-2 Modellierung der Komplexität bei zweispaltigen Relationship Tabellen

Bei Abkehr vom reinen ER-Modell und der Einbeziehung von Fremdschlüsselrelationen und Existenzbedingungen können die erweiterten Komplexitäten des strukturierten ER-Modells, wie in Tab. 4-3 gezeigt, definiert werden.

Komplexität	Fremdschlüsselrelation
1:1	Es ist ein UNIQUE Index über die Fremdschlüsselspalte definiert und ein NOT NULL Constraint
1:*	Es ist kein UNIQUE Index über die Fremdschlüsselspalte definiert, aber ein NOT NULL Constraint
0:1	Es ist ein UNIQUE Index über die Fremdschlüsselspalte definiert, aber kein NOT NULL Constraint
0:*	Es ist weder ein UNIQUE Index über die Fremdschlüsselspalte, noch ein NOT NULL Constraint definiert

Tab. 4-3 Modellierung der Komplexität im SERM

Die Zuordnung der SERM-Komplexitäten in (min,max) Notation zu den ER-Komplexitäten wurde bereits in Kapitel 2.2.2 erläutert (s.a. Abb. 2-2). Mit Hilfe von Tab. 4-2 und Tab. 4-3 lassen sich dann die entsprechenden Zuordnungen festlegen. Reine Relationship-Tabellen enthalten nur die zwei oder mehr Schlüssel der zu verknüpfenden Tabellen. In der AIM Datenbank sind dies alle *Objekttyp*_`Picture` und *Objekttyp*_`Textfile` Relationen. Zusätzlich noch `Engine_Injector`, `Mission_Satellite`, `Rule_Reference`, `Spacestation_Mission`, `Spacestation_Module` und `Subject_Keyword`. Der überwiegende Teil aller Tabellen sind jedoch ER-Tabellen (33 E-Tabellen, 25 R-Tabellen, 153 ER-Tabellen, insgesamt 211 Tabellen).

Die große Anzahl an ER-Tabellen erklärt sich einerseits durch die Datentabellen, die in dieser Auflistung wie ER-Tabellen betrachtet werden, andererseits durch den typischen Aufbau technischer Systeme. In der AIM Datenbank sind die Objekttypen in der Regel technische Systeme und ihre Komponenten. Zum Beispiel teilt sich der Objekttyp `Launcher` in die Subsysteme `Stage`, diese wiederum in `Engine` und `Engine` in die Subsysteme und Komponenten `Gasgenerator`, `Igniter`, `Injector`, `Pump`, `Turbine` und `Valve`. Bei der Modellierung dieser Beziehung genügt es nun nicht zu wissen, dass ein Objekt vom Objekttyp `Launcher` aus bestimmten Stufen besteht, sondern es müssen in aller Regel weitere Angaben gemacht werden, welche dann in der R-Tabelle abgespeichert werden und diese zu einer ER-Tabelle im Sinne des SERM machen. Um bei dem vorgenannten Beispiel zu bleiben, ist es für den Ingenieur wichtig zu wissen, ob es sich bei der durch eine Relation mit dem `Launcher` verbundenen Stufe um die erste, zweite, usw. Stufe handelt.

Alternative Entwürfe

Vor allem während der Bemühungen um eine Integration externer Datenbanken (hauptsächlich bestehende Datenbanken russischer Institute), wurden verschiedene Alternativen zum gewählten Datenmodell diskutiert. Die Alternativen und die einzelnen Argumente sollen hier kurz genannt werden, um die Motive für die Wahl des Datenmodells zusätzlich zu verdeutlichen.

Als eine mögliche Vereinfachung des Datenmodells wurde vorgeschlagen, die Datentabellen nicht mehr zu unterteilen, sondern für jeden Objekttyp nur jeweils eine Datentabelle anzulegen. Das hätte für die SQL-Abfragen den Vorteil, dass nicht geprüft werden muss, in welcher Tabelle sich das gesuchte Attribut befindet. Auch müsste keine Rücksicht darauf genommen werden, ob eine Einheitenspalte (UNIT Spalte) existiert oder eine Standardeinheit ("kg") angenommen wird.

Allerdings handelt man sich dadurch, dass man nur ein Zahlenformat für die Datenspalte festlegen kann, Probleme bei der Speicherung von Datumsformaten ein. Entweder speichert man alle Daten im `CHAR` Format und verzichtet auf alle arithmetischen Funktionen des RDBMS oder man stellt Kalenderdaten als Zahläquivalent dar. Andere Alternativen wären eine zusätzliche Spalte nur für Kalenderdaten

oder eine zusätzliche Tabelle für die Kalenderdaten, womit man aber wieder eine Einteilung vorgenommen hätte. Außerdem kann die Datentabelle für einen Objekttyp bei großen Mengen an Zahlenmaterial sehr groß werden, so dass Suchfunktionen längere Bearbeitungszeiten beanspruchen.

Eine noch weitergehende Vereinfachung wäre die Speicherung aller Objekttypen in einer Tabelle mit einer zusätzlichen Spalte zur Unterscheidung der Klassen. Zusätzlich würden die Eigenschaften ebenfalls in einer einzigen (oder zwei) Tabellen gespeichert werden. Für die Datentabelle gelten dieselben Vor- und Nachteile wie im vorigen Absatz beschrieben. Für die Relationen zwischen den Objekttypen würden sich dann verschiedene R- oder ER-Tabellen anbieten. Allerdings könnte man in diese Tabellen auch unsinnige Relationen eintragen, da die einzelnen Objekttypen nur über einen Eintrag in einer Spalte der Entity-Tabelle unterschieden werden. Bei dieser Konstellation gilt in noch stärkerem Masse, dass bei Anwachsen der Datenmenge und bei Anwachsen der Anzahl der zu speichernden Objekte, die Entity-Tabellen sehr unübersichtlich werden würde. Auch würden Anfragen an die Datenbank sehr lange Verarbeitungszeiten benötigen, da für alle Relationen immer die sehr große Entity-Tabelle konsultiert werden müsste.

4.3 Beschreibung der E-Tabellen und der Relationen zwischen den Objekten

Einen Überblick über die zum gewählten Datenmodell gehörenden RDBMS-Objekte (Tabellen, Indizes) geben Tab. 4-4 und Tab. 4-5, sowie die im Anhang aufgeführten Übersichten. Tab. 4-4 zeigt zunächst alle Objekttypen der AIM Datenbank.

Tabelle	Beschreibung
Address	Adressen (von Firmen, Organisationen und Personen)
Engine	Triebwerke
Fairing	Nutzlastverkleidungen der einzelnen Träger
Folder	Ordner der Fachgebietsbibliothek in denen sich die Quellen (Reference) befinden
Gasgenerator	Gasgeneratoren von Triebwerken
Igniter	Zünder von Triebwerken
Injector	Einspritzsysteme von Triebwerken
Keyword	Stichworte zu den Quellen (Reference)
Launch	Starts von Trägersystemen und Missionen
Launcher	Trägersysteme
Launchsite	Startorte
Mission	Missionen die gestartet wurden
Module	Module von Raumstationen
Nation	Nationen, Länder und Staatenbünde
Orbit	Zielorbits von Raketenstarts und Umlaufbahnen von Satelliten
Organization	Raumfahrtfirmen und -organisationen
Person	Autoren, Astronauten, Kosmonauten, Firmen- und Organisationsangehörige
Picture	Bilder und Zeichnungen (und auch einige Filme) zu den Objekten der Datenbank
Propellant	Treibstoffkombinationen, die bei Raketentriebwerken verwendet werden
Pump	Pumpen von Triebwerken
Reference	Bücher, Zeitschriften, Artikel, Aufsätze, Vorträge, etc.
Rule	Regeln, die in der Regelbasis der Expertensystemkomponente verwendet werden
Satellite	Satelliten, die bei bestimmten Missionen gestartet wurden
Spacestation	Raumstationen
Stage	Trägerstufen
Subject	Sachgebiete zur Einordnung der Stichwörter in Kategorien
Substance	Chemische Elemente und Verbindungen (u.a. Treibstoffe)
Subsystem	Subsysteme von Raumstationen und Modulen
Textfile	Texte zu den Objekten der Datenbank (hauptsächlich Träger, Stufen und Triebwerke)
Trajectory	Aufstiegsbahnen von Trägerraketen
Turbine	Turbinen von Triebwerken
Valve	Ventile, die in Triebwerken verwendet werden

Tab. 4-4 Objekte der AIM Datenbank (Entity-Tabellen)

Ein Objekttyp oder eine Klasse von Objekten wird in der AIM Datenbank durch eine Entity-Tabelle repräsentiert. Ein Objekt der Raumfahrt wird in diesen Tabellen durch eine Zeile eindeutig definiert. Die Subsysteme dieser Objekte sind wiederum Objekte, die in anderen Entity-Tabellen gespeichert sind. Aus welchen Subsystemen ein Objekt besteht, wird in der Datenbank über die Relationen gespeichert. Das hat den Vorteil, dass die Eigenschaften eines Objekts nur einmal gespeichert werden müssen, selbst dann, wenn dieses Objekt in mehreren anderen Objekten als Subsystem verwendet wird. Dies bedeutet aber auch, dass wenn sich eine Eigenschaft eines Subsystems so ändert, dass ein neues Objekt definiert werden muss (z.B. der Brennkammerdruck eines Triebwerks wird erhöht und das Triebwerk wird als neues Objekt in die Datenbank eingefügt), dann muss für alle Systeme in denen dieses Subsystem verwendet wird, ebenfalls ein neues Objekt eingeführt werden. In dem genannten Beispiel muss eine neue Stufe und ein neuer Träger eingetragen werden. Für den Datenbankadministrator bedeutet das, bei jeder technischen Änderung zu entscheiden, ob ein neues Objekt definiert werden soll oder nicht. Im Extremfall hieße das, wenn sich die Drehzahl einer Turbopumpe ändert und die Pumpe als neues Objekt eingegeben wird, dann müsste auch ein neuer Träger angelegt werden. Da Raumfahrtsysteme ständig weiterentwickelt werden, ist bei jeder Änderung abzuwägen, ob die Änderung so signifikant ist, dass ein neues Objekt eingeführt wird oder nicht.

Entity-Tabelle	Relation mit	1)	2)	zusätzliche Eigenschaften
Address	Person Nation	Organization	-	-
Attribute	-	-	-	-
Engine	Gasgenerator Igniter Injector Pump Stage Turbine Valve	X	X	Emission Reliability
Fairing	Launcher Launch	X	X	-
Folder	Reference	-	-	-
Gasgenerator	Engine	X	X	-
Igniter	Engine	X	X	-
Injector	Engine	X	X	-
Keyword	Reference	-	-	-
Launch	Launcher Orbit Mission Reference	X	Cost	Failure
Launcher	Fairing Stage Launch	X	X	Failure Payload Reliability
Launchsite	Launcher Trajectory Launch	X	X	Payload
Mission	Orbit Person Satellite Spacestation	X	Alias Category Cost Date	EVA Failure
1) Relation mit Organization, Picture, Textfile 2) Verbindung zu den Eigenschaftentabellen Alias, Category, Cost, Date, Geometry, Mass, Performance				

Tab. 4-5a Relationen zwischen den Entity-Tabellen

Entity-Tabelle	Relation mit	1)	2)	zusätzliche Eigenschaften
Module	Spacestation	X	X	-
Nation	Address Person	X	Alias	-
Orbit	Launcher Trajectory Launch Mission	Picture Textfile		GEO EVA Payload
Organization	diverse	Picture Textfile	Alias Category	Address
Person	Address Mission Nation Reference	X	Alias Category	EVA
Picture	diverse	Textfile	-	-
Propellant	Engine Substance	X	Alias Category Cost Performance	Burnrate Emission
Pump	Engine	X	X	-
Reference	Folder Keyword Rule	Organization Textfile	Category Date	-
Rule	Reference	Picture Textfile	Category	Condition Result
Satellite	Mission	X	X	-
Spacestation	Mission Module Nation Subsystem	X	X	-
Stage	Launcher Engine	X	X	Reliability
Subject	Keyword	-	-	-
Substance	Propellant	X	Alias Category Cost	Composition Cubicalexpansion Density, Enthalpy Heatcapacity Heatconductivity Isentropicexponent Property Surfacetension Vaporpressure Viscosity
Subsystem	Spacestation	X	X	-
Textfile	diverse	-	-	-
Trajectory	Launcher Launch Orbit	Picture Textfile	-	Events Vectors
Turbine	Engine	X	X	-
Valve	Engine	X	X	-
1) Relation mit Organization, Picture, Textfile 2) Verbindung zu den Eigenschaftentabellen Alias, Category, Cost, Date, Geometry, Mass, Performance				

Tab.4-5b Relationen zwischen den Entity-Tabellen (Fortsetzung)

Allgemeine Tabellen

Mit allgemeinen Tabellen sind hier Entity-Tabellen gemeint, die einer Vielzahl anderer Objekte der Datenbank zugeordnet werden können. Diese Tabellen dienen nicht dazu den konstruktiven Aufbau von technischen Systemen zu erfassen, sondern sind eine Verbindung zu zusätzlichen Informationen, die sich auf nahezu alle Objekte der AIM Datenbank erstrecken. Diese allgemeinen Tabellen umfassen die alternativen Bezeichnungen für die einzelnen Objekte (Alias), die Einteilung der Raumfahrtobjekte nach verschiedenen Gesichtspunkten (Category) und die Zuordnung zu Objekten, die aufgrund ihres Umfangs nicht Bestandteil der Datenbank sind (Picture, Textfile). Zusätzlich soll hier noch auf das Problem der „Nationalität“ von Raumfahrtobjekten eingegangen werden.

Alias, Category

Nahezu alle Entity-Tabellen besitzen noch eine Relation mit den Tabellen *Objekttyp*_Alias und *Objekttyp*_Category (s.a. Tab. 4-5). Da in der Literatur die Bezeichnung für Objekte der Raumfahrt nicht immer einheitlich geschrieben werden, teilweise werden auch verschiedene Namen für ein und dasselbe Objekt benutzt, ist eine klare Zuordnung zwischen Name und Objekt nicht möglich. Da der Benutzer diese aber auch unter dem ihm bekannten Namen, der eventuell nicht identisch mit dem in der Datenbank ist, finden soll, wurden die Alias-Tabellen eingeführt, in denen beliebig viele alternative Bezeichnungen für ein Objekt gespeichert werden können. In der Macintosh Version der Abfragesoftware ist auch die Speicherung von Namen mit kyrillischen Buchstaben möglich, da durch die Kooperation mit russischen Universitäten ein großer Bestand an Daten über die russische Raumfahrt integriert werden konnte. Den russischen Gastwissenschaftlern wird die Benutzung der Datenbank dadurch erheblich erleichtert, da es für einige Objekte russische Bezeichnungen gibt, die im Westen nicht verwendet werden.

Die Relation *Objekttyp*_Category erlaubt die beliebige Einteilung der Raumfahrtobjekte nach unterschiedlichen Kriterien. Sie wird zum Beispiel benutzt, um das in Kap. 0 beschriebene Schichtenmodell innerhalb der Datenbank zu speichern, aber auch andere Einteilungen können gespeichert werden. So können z.B. die Triebwerke in Flüssigtreibstoff- und Festtreibstofftriebwerke und die Trägersysteme in Wiederverwendbare und Nicht-wiederverwendbare eingeteilt werden. Durch die Verwendung der comp(0,*) Komplexität können für ein Objekt mehrere Kategorisierungen (z.B. nach unterschiedlichen Kriterien) angewandt werden. Dadurch kann sowohl die Ausgabe von Ergebnissen sinnvoll strukturiert werden (wenn die Abfragesoftware dies unterstützt) als auch die Suche nach Objekten über Kategorien an Stelle von Namen durchgeführt werden (wie mit dem Schichtenmodell geschehen).

Picture, Textfile, (Movie)

Die Entity-Tabellen Picture, Textfile und in Zukunft auch Movie enthalten Verweise auf Daten, die nicht innerhalb der ORACLE Datenbank gespeichert sind. Vielmehr sind die Dateinamen von Text- und Bilddateien zusammen mit einigen zusätzlichen Informationen abgelegt. Die Verbindung zwischen den Raumfahrtobjekten und diesen Dateien wird über R-Tabellen (*Objekttyp*_Picture und *Objekttyp*_Textfile) gespeichert. Der Benutzer erfährt bei einer Abfrage nur, dass eine Datei zu dem entsprechenden Objekt existiert und wo sie gespeichert ist. Sollen diese externen Informationen dem Benutzer zugänglich gemacht werden, so muss die Abfragesoftware die in den Tabellen gespeicherten Informationen auswerten und die Dateien anzeigen. Das WWW Interface benutzt diese Informationen, um die Texte und Bilder bei der Generierung der HTML-Seiten zu integrieren.

Die Komplexität der Relationen erlaubt eine Zuordnung beliebig vieler Bilder und Texte zu einem Raumfahrtobjekt, aber auch die Zuordnung eines Bildes oder Textes zu mehreren Raumfahrtobjekten. Dieser Umstand wirft bei der Speicherung der Dateien gewisse Probleme auf. Die Dateien befinden sich auf dem WWW Server in verschiedenen Unterverzeichnissen, die nach den Objekttypen benannt sind. Dies geschieht der Übersicht halber, aber auch, um eine eventuelle Begrenzung der Dateieinträge pro Verzeichnis zu umgehen. Durch die M:N-Zuordnung müssten sich manche Dateien in mehreren

Verzeichnissen befinden, was aber aus Platzgründen nicht sinnvoll ist. Auf dem Apple Macintosh kann dieses Problem durch Aliasdateien gelöst werden. Bei einer eventuellen Portierung des WWW Servers auf ein anderes Betriebssystem muss dieser Umstand berücksichtigt werden.

Nation, Organization

Ein besonderes Problem für die Tabellenstruktur erwächst aus der in der Raumfahrt üblichen Einteilung der Raumfahrtobjekte nach Nationen. So spricht man von amerikanischen oder russischen Trägersystemen oder Satelliten. Es lag deshalb nahe, bei allen Raumfahrtobjekten, die technische Systeme repräsentieren, über eine Relation *Objekttyp*_Nation diese Information zuzuordnen. Dabei tritt natürlich bei der europäischen Rakete Ariane das Problem auf, dass mehrere Nationen an der Entwicklung der Rakete beteiligt sind. Eine elegante Lösung wäre nun mehrere Nationen pro Objekt zuzulassen und entsprechend einzutragen. Wenn man allerdings eine Übersicht aller Trägersysteme nach Nationen gruppiert haben möchte, wird die Ariane mehrfach auftreten. Dies kann vermieden werden, wenn man Europa zur Nation erklärt und in die Tabelle Nation einträgt. Ein weiteres Problem, welches während der Arbeit an der Datenbank auftrat war der Zerfall der Sowjetunion. Alle ehemals sowjetischen Trägersysteme waren plötzlich russische, kasachische, ukrainische oder GUS Träger geworden. Würde man einfach in der Tabelle Nation die UdSSR in Russland oder GUS umbenennen, würden auch alle nicht mehr gebauten Träger zu russischen bzw. GUS Trägern werden, was aber nicht den historischen Tatsachen entspricht. Man muss hier sehr aufpassen und im Einzelfall unterschiedliche Entscheidungen treffen. Eine Absicherung über eine intelligente Tabellenstruktur wäre nur möglich, wenn in der Relation *Objekttyp*_Nation eine zusätzliche Spalte eingefügt wird, die angibt, in welchem Zeitraum die Zuordnung gültig ist. Diese müsste natürlich Teil des Schlüssels sein und würde Abfragen, in denen diese Relation verwendet wird, unnötig kompliziert machen.

Eine bessere Lösung wäre der Wegfall der *Objekttyp*_Nation Relation und die Feststellung der Nationalität über Regeln, die sich z.B. auf die Hersteller abstützen. Es werden jedem Raumfahrtobjekt eine oder mehrere Firmen oder Organisationen über die *Objekttyp*_Organization ER-Tabellen zugeordnet, die jeweils unterschiedliche Aufgaben, repräsentiert durch die *Objekttyp*_OrgTask Spalte, erfüllen. Jetzt kann man die Nationalität eines Trägersystems zum Beispiel an der Nationalität des „Main Designer“ oder „Main Manufacturer“ festmachen. Diese Information ist entweder über die Relation Organization_Nation oder über die Adresse des Hauptsitzes der jeweiligen Organisation oder Firma Organization_Address (die E-Tabelle Address enthält eine Nation_ID Spalte) zu beschaffen. Probleme die bei dieser Lösung auftreten sind z.B. Lizenzbauten von Trägern (amerikanische Trägersysteme sind in Lizenz in Japan gebaut worden) aber auch der Kauf und Start fremder Träger (Italien hat mehrere Scout Raketen von den USA gekauft und von einem eigenen Startplatz vor der afrikanischen Küste gestartet).

Ausgewählte Raumfahrtobjekte

Die wichtigsten Objekttypen der AIM Datenbank sind die Klassen Launcher, Stage und Engine. Da sich das Fachgebiet Raumfahrzeugtechnik hauptsächlich mit Raumtransportsystemen beschäftigt, wurde besonderer Wert auf die Speicherung von Daten aus diesem Bereich der Raumfahrt gelegt. Erst in zweiter Linie werden die transportierten Objekte, also Missionen/Satelliten und Raumstationen betrachtet.

Launcher

Der Objekttyp Launcher bildet zusammen mit dem Objekttyp Engine den zentralen Bereich der AIM Datenbank. Ausgehend von der Entity-Tabelle Launcher und den bereits beschriebenen Datentabellen (Kosten, Massen, Geometrie, etc.) wurde versucht alle Informationen der Diskurswelt in einem konsistenten Datenschema abzubilden. Hierbei kann zwischen zwei grundsätzlichen Bereichen unterschieden werden: zunächst das technische System mit allen Subsystemen bis hin zu einzelnen Komponenten, dann der Bereich der Trägersystemstarts und -aufstiegsbahnen.

Im technischen Bereich sind alle relevanten Daten, die den gesamten Träger betreffen, in den entsprechenden Datentabellen abgelegt. Durch die Aufteilung eines Raumtransportsystems in einzelne Stufen sind die detaillierten technischen Daten auf die Datentabellen des Objekttyps `Stage` und über die Verbindung `Launcher_Stage` → `Stage_Engine` auch zum Teil auf den Objekttyp `Engine` verteilt. Eine weitere Komponente heute üblicher Raketen ist die Nutzlastverkleidung. Da bei manchen Trägern verschiedene Nutzlastverkleidungen benutzt werden, wurde eine eigene Relation (`Launcher_Fairing`) und ein eigener Objekttyp (`Fairing`) geschaffen. Diese Relation gibt an, welche Nutzlastverkleidung von welchem Träger benutzt wird. Es fehlt die Information, bei welchem speziellen Start des Trägers welches Fairing zum Einsatz kam. Da diese Information in der Tabelle `Launch_Launcher` zu finden ist, liegt hier eine Redundanz vor. Die Information welche Nutzlastverkleidungen von welchem Träger benutzt werden, sollte aus der Tabelle `Launch_Launcher` entnommen werden und die Tabelle `Launcher_Fairing` sollte gelöscht werden.

Die Nutzlastkapazität eines Trägersystems, die man eigentlich in der Datentabelle `Launcher_Performance` erwarten würde, ist noch von mehreren anderen Faktoren abhängig und daher in einer eigenen Tabelle (`Launcher_Payload`) erfasst. Die Nutzlastkapazität eines Trägersystems variiert mit Zielorbit, Startort und gegebenenfalls auch mit der geflogenen Aufstiegsbahn. Da man aber davon ausgehen kann, dass jeder Träger für einen bestimmten Orbit eine nutzlastoptimale Aufstiegsbahn fliegt, wurde nur die Abhängigkeit vom Startort und Zielorbit berücksichtigt. Selbst wenn keine optimierte Bahn geflogen wird, so kann man doch davon ausgehen, das für bestimmte Startort/Zielorbit-Kombinationen immer die gleiche Aufstiegsbahn geflogen wird, so dass eine funktionale Abhängigkeit gegeben ist.

Die Information, welche Aufstiegsbahn geflogen wird, ist in der ER-Tabelle `Launch_Orbit` enthalten. Dort wird eine Verbindung zwischen den Entity-Tabellen `Orbit`, `Trajectory` und `Launch` hergestellt. Über `Launch_Launcher` werden `Launcher`, `Launchsite` und `Fairing` zugeordnet. Die Verbindung der einzelnen Tabellen über `Launch_Launcher` ist sehr ungünstig, denn es wäre besser alle diese Zuordnungen in einer Tabelle (`Launch`) zu konzentrieren. Dies war auch der ursprüngliche Entwurf, jedoch war in der Literatur bei verschiedenen Starts die Zuordnung eines Trägers zu einem Startdatum zum Teil widersprüchlich (insbesondere bei militärischen oder sowjetischen Starts). Daher schien die Möglichkeit der Zuordnung mehrerer Träger (aus verschiedenen Quellen) zu einem Start, bei gleichzeitiger Festlegung eines wahrscheinlichsten Trägers, eine günstige Lösung zu sein. Dadurch bleiben zwar die Informationen aus den verschiedenen Quellen erhalten, jedoch handelt man sich immer dann immense Probleme ein, wenn eine eindeutige Zuordnung Träger↔Start erforderlich ist. Da dies in der Diskurswelt aber immer erwartet wird, sollte die Aufteilung der Informationen auf mehrere Tabellen wieder rückgängig gemacht werden, zumal viele der Mehrfacheintragungen in der Tabelle `Launch_Launcher` im Laufe der Zeit eindeutig identifiziert werden konnten.

Die Aufstiegsbahndaten (Geschwindigkeits- und Ortsvektor über der Zeit) befinden sich in der Tabelle `Trajectory_Vectors`. Zusätzlich sind noch die bahnrelevanten Ereignisse (Triebwerkszündung, Stufentrennung, etc.) in der Tabelle `Trajectory_Events` erfasst.

Die weltweit verteilten Startplätze sind in der Tabelle `Launchsite` zu finden. Die auf den Objekttyp anwendbaren Datentabellen wurden implementiert. In der Regel besteht ein Weltraumbahnhof aus mehreren Startanlagen (Launchpads), für die jedoch die gleichen Attribute gelten wie für den gesamten Startplatz. Daher wurde kein neuer Objekttyp eingeführt und die Launchpads befinden sich in der Tabelle `Launchsite`. Um eine Zuordnung der Startanlagen zu den Startplätzen zu ermöglichen, müsste bei Bedarf eine zusätzliche Relation `Launchsite_Launchsite` erstellt werden.

Stage

Die E-Tabelle `stage` stellt das Bindeglied zwischen den Objekten `launcher` und `engine` dar. Die Relation `Launcher_Stage` legt fest, dass ein Trägersystem aus 1 bis n Stufen bestehen kann. Eine Stufe kann in mehreren Trägern verwendet werden. Allerdings kann eine Stufe auch mehrfach bei einem

Träger verwendet werden. Dieser Fall tritt eigentlich nur bei der Verwendung von Boostern auf. So besteht z.B. die 0. Stufe der Ariane 42L aus zwei Flüssigtreibstoffboostern. Die Stufennummer wird in der Spalte `Launcher_StageNo` und die Anzahl in der Spalte `Launcher_StageAmount` gespeichert. Dies entspricht einer geometrischen Einteilung der Stufen quasi nach Baugruppen. Für den Raumfahrtingenieur insbesondere im Hinblick auf Aufstiegsbahnen ist jedoch eine Einteilung nach der Raketengrundgleichung interessanter. Hier wird eine Stufe durch sprunghafte Änderungen in der Masse und dem spezifischen Impuls gekennzeichnet. Dies ist in der Regel der Fall, wenn sich Teile von der Rakete trennen (z.B. Abtrennung verbrauchter Stufen). Dies wird durch die Spalte `Launcher_StageBlockNo` berücksichtigt. Abb. 4-3 verdeutlicht dies am Beispiel der Delta 7925. Die dort eingetragene Zahl gibt die Reihenfolge gleichzeitig abgeworfener Stufen an. Bei der Delta Rakete, die 9 Booster verwendet, werden zunächst 6 und dann 3 Booster gezündet und abgeworfen.

Launcher

ID	name	Family	Version	# stg	# boo	lift-off	landir
41	Delta 7925	Delta II	II	3	9	VTO	no

Launcher_Stage

rowID	Block Number	Launcher_ID	Stage_ID	Number	Amount
000007FB.0030.0003	1	41	50	0	3
00000802.000D.0003	2	41	50	0	3
00000802.0019.0003	3	41	256	0	3
000007FB.0031.0003	4	41	51	1	1
000007FB.0032.0003	5	41	45	2	1
000007FB.002F.0003	6	41	46	3	1

2,4

Abb. 4-3 Auszug aus den Tabellen `Launcher_Stage` und `Launcher`

Engine

Der Objekttyp `Engine` stellt neben den Trägersystemen einen weiteren Hauptbestandteil der AIM Datenbank dar. Zunächst waren alle Informationen in der E-Tabelle und den entsprechenden Datentabellen gespeichert. Allerdings stellte sich bald heraus, dass im Bereich der Triebwerke einzelne Komponenten in mehreren Triebwerken verwendet wurden. Dies liess eine Normalisierung und somit eine Aufteilung der Daten auf mehrere Tabellen sinnvoll erscheinen. Alle Subsysteme bis auf Brennkammer und Düse wurden zu Entities und damit zu neuen Objekttypen mit eigenen Entity- und Datentabellen. Die Verbindung zum Triebwerk wird über ER-Tabellen bewerkstelligt. Da eine weitere Zunahme der Standardisierung im Triebwerksbau zu erwarten ist, wiegt die mit der Normalisierung erreichte Redundanzfreiheit, den geringfügig größeren Aufwand bei der Abfrage und Manipulation der Daten weitgehend auf.

Triebwerke, die in Satelliten oder Raumstationen zur Lageregelung bzw. Bahnregelung verwendet werden, können durch Einträge in die ER-Tabellen `Satellite_Engine` bzw. `SpaceStation_Engine` den jeweiligen Objekten zugeordnet werden.

Probleme im Bereich der Triebwerke entstehen hauptsächlich dadurch, dass so unterschiedliche Triebwerke wie Feststoffbooster und elektrische Triebwerke mit den in der Raumfahrt hauptsächlich verwendeten Flüssigtreibstofftriebwerken innerhalb einer Datenstruktur gespeichert werden. Bisher konnten jedoch aufgrund der extrem flexiblen Organisation der Daten alle Attribute erfasst werden. Ein weiteres Problem entsteht durch die vor allem in Russland übliche Konstruktionsweise der Triebwerkscluster (mehrere zum Teil unterschiedliche Triebwerke teilen sich einen oder mehrere Turbopumpensätze). In der AIM Datenbank werden solche Cluster in einigen Fällen wie ein Triebwerk behandelt, in anderen Fällen wie einzelne Triebwerke. Eine einheitliche Lösung wäre die Behandlung als ein Triebwerk mit einer zusätzlichen Tabelle (z.B. `Engine_Cluster`), in der die Komponenten (Einzeltriebwerke, Turbopumpen, etc.) mit weiteren Eigenschaften eingetragen werden.

Propellant, Substance

Der ursprüngliche Zweck der E-Tabelle `Propellant` war die Erfassung der Triebwerksemissionen der verschiedenen Trägersysteme für eine Studie zur Belastung der Atmosphäre durch die Raumfahrt /1/, /25/. Für jede Treibstoffkombination wurden, in Abhängigkeit von Brennkammerdruck, Mischungsverhältnis und Flächen- oder Druckverhältnis, die am Düsenende entstehenden Substanzen berechnet. Alle entstehenden Stoffe wurden in einem Objekttyp `Substance` erfasst und die Rechenergebnisse in der ER-Tabelle `Propellant_Emission` abgelegt. Damit ist es möglich, einzelnen Treibstoffkombinationen typische Emissionen zuzuordnen, so dass über die `Launcher_Stage`→`Stage_Propellant` Verbindung für jeden Träger typische Emissionsprodukte ermittelt werden können. Um die Genauigkeit der Werte zu steigern, kann die Tabelle `Engine_Emission` benutzt werden, in der die Werte für den Brennkammerdruck, Mischungsverhältnis und Flächen- oder Druckverhältnis vom jeweiligen Triebwerk benutzt werden, um die für dieses Triebwerk typischen Verbrennungsprodukte am Düsenende zu ermitteln.

Später wurde die Tabelle `Substance` für die Speicherung von Treibstoffen benutzt, deren Eigenschaften in zusätzlichen Tabellen gesichert wurden (`Substance_Density`, `Substance_Enthalpy`, etc.). Zusätzlich wurde im Hinblick auf die Zusammensetzung von Festtreibstoffen eine Relation `Propellant_Substance` eingeführt, in der die Komponenten (d.h. die vorhandenen Substanzen oder Treibstoffkomponenten) mit einem prozentualen Anteil, Aggregatzustand, etc. eingetragen werden können. Dies kann natürlich auch für zusätzliche Informationen bei den übrigen Treibstoffkombinationen genutzt werden.

Mission, Satellite

Die Tabelle `Mission` gibt an, welche Missionen (Satelliten) bei einem Trägerraketenstart (`Launch`) befördert wurden. Da jede Mission nur einmal starten kann, bei einem Start aber mehrere Missionen gestartet werden können, wird die `Launch_ID` innerhalb der E-Tabelle `Mission` gespeichert. Da einige Satellitentypen in den USA mittlerweile in Serie produziert werden (insbesondere große Nachrichtensatelliten z.B. INTELSAT und Spionagesatelliten), wurde keine direkte Relation zwischen Start und Satellit hergestellt, sondern durch die Relation `Mission_Satellite` berücksichtigt, dass ein Satellitentyp bei verschiedenen Missionen starten kann. Wird kein Satellit, sondern eine Raumstation gestartet, findet sich der entsprechende Eintrag in der R-Tabelle `Mission_Spacestation`.

Die Datentabellen `x_Geometry`, `x_Mass` und `x_Performance` machen bei Missionen keinen Sinn. Kosten, Datumsangaben, alternative Bezeichnungen und Missionskategorien werden dagegen berücksichtigt. Die aktuellen Orbitparameter einer Mission können über die Relation `Mission_Orbit` in der Tabelle `Orbit` gefunden werden. Bei bemenschten Missionen sind die Personen und ihre Aufgabe während der Mission in der Tabelle `Mission_Person` bzw. der Spalte `Mission_PersFunction` gespeichert. Sollten während einer Mission Weltraumspaziergänge stattfinden, werden die entsprechenden Informationen in der Tabelle `Mission_EVA` gespeichert. Bei fehlgeschlagenen Missionen können nähere Beschreibungen der Fehlerursache der Tabelle `Mission_Failure` entnommen werden.

Spacestation, Module, Subsystem

Die Raumstationen werden isoliert betrachtet und besitzen nur wenige Verbindungen zu den anderen Objekten. Die Relation `SpaceStation_Mission` gibt an, mit welchen Missionen die Raumstation in den Orbit gebracht wurde. `SpaceStation_Nation` bzw. `SpaceStation_Organization` stellen die üblichen Informationen über Entwickler, Hersteller und Betreiber zur Verfügung. Eine Raumstation kann aus mehreren Modulen bestehen (`Module, Spacestation_Module`) und verschiedene Subsysteme wie Bordrechner, Lebenserhaltung, Experimentenracks, etc. enthalten (`Subsystem`, `Spacestation_Subsystem`). Da sich die Subsysteme meist auf die gesamte Station beziehen, wurde auf eine Zuordnung `Module_Subsystem` verzichtet. Diese kann jedoch bei Bedarf eingerichtet werden.

Reference, Folder, Subject, Keyword, Person

Die Verwendung der Tabelle `Person` ist sehr vielfältig. Hier werden hauptsächlich Autoren von Büchern und Artikeln (`Reference_Person`), aber auch Astronauten und Kosmonauten (`Mission_Person`) sowie leitende Angehörige von Firmen und Organisationen (`Person_Organization`) erfasst. Personen können mit mehreren Adressen (`Person_Address`) und Nationen (`Person_Nation`) verknüpft werden. Bei Astronauten und Kosmonauten können zusätzlich zu den geflogenen Missionen eventuelle EVAs erfasst werden (`Mission_EVA`).

Die Hauptaufgabe der oben genannten Tabellen ist jedoch die Verwaltung der Fachgebietsbibliothek. Die Tabelle `Reference` speichert alle Titel und zusätzlichen Angaben zu den in der Bibliothek vorhandenen Büchern, Studien-, Diplom- und Doktorarbeiten, Artikeln, Paper, etc. Die Autoren einer Quelle werden über die Relation `Reference_Person` bereitgestellt. Dabei kann durch die Spalte `Person_Task` zwischen Autoren, Verlegern, Editoren etc. unterschieden werden. Zwischen den Autoren kann mit der Spalte `Reference_PersonNumber` eine Reihenfolge festlegt werden.

Zu jeder Quelle kann eine Organisation oder Firma (`Reference_Organization`) zugeordnet werden. Über `Reference_TextFile` kann der Text der Quelle als Datei abrufbar gemacht werden. Den Quellen sind Stichworte zugeordnet (`Reference_Keyword`), die wiederum wegen ihrer großen Zahl und der damit verbundenen Unübersichtlichkeit in Kategorien eingeteilt sind (`Subject_Keyword`). Sowohl `Subject` als auch `Keyword` sind in deutsch und englisch verfügbar. Die Literatur ist in nummerierte und betitelte Ordner sortiert, deren Nummer und Titel in der Tabelle `Folder` gespeichert sind. Der Standort der Quellen innerhalb der Bibliothek wird über `Reference_Folder` gespeichert.

Rule, Attribute

Die E-Tabelle `Rule` und die zugehörigen ER-Tabellen (`Rule_Condition` und `Rule_Result`) unterstützen die Erklärungskomponente des Expertensystemteils und werden in Kap. 6.5 ausführlich beschrieben.

Die Tabelle `Attribute` hat innerhalb der Datenbank keinerlei Funktion, bietet aber eine Möglichkeit, Erklärungen und Übersetzungen zu den Attributen anzubieten. Es bestehen zur Zeit keine Verbindungen zu anderen Tabellen der Datenbank.

5 Komponente zur Feststellung der Lücken im Faktenwissen

Aufgrund der gewählten Datenstruktur des Faktenwissens ist es nicht möglich ohne größeren Aufwand Lücken im Faktenwissen zu erkennen. Das Expertensystem weiß nicht, wo sein Faktenwissen lückenhaft ist, da eine starre Zuordnung von Attributen zu den Objekten, aus den in den vorhergehenden Abschnitten erläuterten Gründen, vermieden wurde. Die Attribute werden nicht durch Spalten einer Tabelle repräsentiert, sondern sind Einträge in einer Attributspalte einer Tabelle. Dadurch kann jedem Objekt eine beliebige Kombination von Attributen zugeordnet werden. Außerdem kann jedes beliebige Attribut in die Datenbank aufgenommen werden.

Ein menschlicher Experte weiß jedoch, dass für bestimmte Objekte nur bestimmte Attribute zugelassen sind. In der AIM Datenbank, die das Faktenwissen für das Expertensystem enthält, ist eine gewisse Einschränkung in der Attributzuordnung zu den Objekten durch die Aufteilung der Datentabellen gegeben (z.B. stehen alle Trägerattribute in den Tabellen `Launcher_Geometry`, `Launcher_Mass`, `Launcher_Performance`, etc.). Diese Aufteilung ist jedoch bei weitem nicht ausreichend. Deshalb muss ein anderer Weg zur Lösung dieses Problems gefunden werden.

Wie bereits erwähnt, weiß ein menschlicher Experte, welche Attribute zu einem Objekt gehören und welche nicht. Zusätzlich hat er durch seine Erfahrung ein Gefühl dafür entwickelt in welchem Bereich sich ein Attributwert für ein bestimmtes Objekt oder für eine ganze Klasse von Objekten in der Regel bewegt. Um dies durch ein Expertensystem nachzubilden muss die Einteilung der Objekte in Klassen entweder fest vorgegeben oder dynamisch vorgenommen werden.

Hat man eine Klassenhierarchie erzeugt und sind jeder Klasse bestimmte Attribute zugeordnet, dann kann man durch Einordnung des fraglichen Objektes in diese Hierarchie diejenigen Attribute bestimmen, die das Objekt besitzen muss. Ein Listenvergleich mit den in der Datenbank bereits vorhandenen Attributen erzeugt eine Liste der fehlenden Attribute und damit der für dieses Objekt bestehenden Lücken im Faktenwissen.

5.1 Formen der Einteilung von Raumfahrtobjekten

Da es sich bei dem Expertensystem auch um ein Informationssystem der Raumfahrt handelt ist es naheliegend Klassifikationen, welche in der Raumfahrt üblich und somit einem größeren Personenkreis bereits bekannt sind, zu verwenden. Deshalb wird im Folgenden eine Auswahl an Klassifikationen kurz vorgestellt.

5.1.1 Objektklassen auf Basis von Attributwerten

Sehr häufig trifft man in der Raumfahrt auf Einteilungen, die auf der Ausprägung bzw. dem Wert eines bestimmten Attributes Einteilungen vornehmen. In den Skripten (/27/, /28/) des Fachgebiets Raumfahrzeugtechnik werden typische Beispiele für die Einteilung von Raketentriebwerken und Transportsystemen genannt. Eine Typeneinteilung der Raumfahrtsysteme richtet sich zum Beispiel nach

- ihrer Funktion oder Aufgabe,
- und dem Einsatzort.

Raumfahrtantriebe sind fast ausschließlich Strahlantriebe. Die Antriebe lassen sich nach der Art der Primärenergie, der Energiewandlung, und des Ursprungs der Strahlmasse unterteilen. Es werden auch Einteilungen nach folgenden Attributwerten vorgenommen (s.a. Tab. 5-1):

- Zahl der Treibstoffkomponenten
- Aggregatzustand der Treibstoffe
- Spezifischer Impuls
- Lagerfähigkeit

Zahl der Komponenten Jeder Stoff, der einen eigenen Tank braucht, ist eine Komponente	Monergole Diergole Triergole
Aggregatzustand der Treibstoffe Nur fest und flüssig	Flüssigkeitsantriebe Feststoffantriebe Hybrid- (Lithergol-)antriebe
Spezifischer Impuls Hier gewichtsspezifischer Impuls bei Standardentspannungsverhältnis pc:pe = 68	Niederenergetische Isp 68:1 < 270 s Mittelenergetische Isp 68:1 270 – 330 s Hochenergetische Isp 68:1 > 330 s
Lagerfähigkeit Nach Übereinkunft definierte Bereiche	Erdlagerfähige keinPhasenwechsel zwischen ca. –40°C bis +80°C Raumlagerfähige kein Einfrieren und niedriger Dampfdruck bei kalter Lagerung Kryogene verflüssigte Gase

Tab. 5-1 Einteilung der chemischen Antriebe nach verschiedenen Gesichtspunkten /27/

Auch eine Einteilung nach Art der Energieerzeugung im Arbeitsgas ist möglich (s. Abb. 5-1). Alle diese Einteilungen sind für den hier gedachten Zweck nicht geeignet, da sie nur auf einem einzigen Attributwert oder einer Kombination von Attributwerten beruhen. Eine Zuordnung von Attributen zu den einzelnen Klassen ist nicht möglich. Dennoch sind solche Einteilungen weit verbreitet und in späteren Abschnitten wird darauf eingegangen, wie man die weite Verbreitung dieser Einteilungen für den benutzerfreundlichen Zugriff auf die Datenbestände des Informationssystems ausnutzen kann.

Bei keinem der genannten Klassifizierungsbeispiele ist eine klare Zuordnung von bestimmten Attributen gegeben. Die Einteilungen dienen der Zusammenfassung von verschiedenen Antrieben und Antriebskonzepten zu logischen Gruppen. Die Zugehörigkeit zu einer Gruppe lässt jedoch keinerlei Rückschlüsse auf den Aufbau des Systems zu. Es können zwar teilweise Subsysteme identifiziert werden, und mit ihnen die entsprechenden Attribute, aber es handelt sich immer nur um stichpunktartige Definitionen. Eine vollständige Bestimmung mit allen Subsystemen und Komponenten ist an Hand der gezeigten Beispiele nicht möglich.

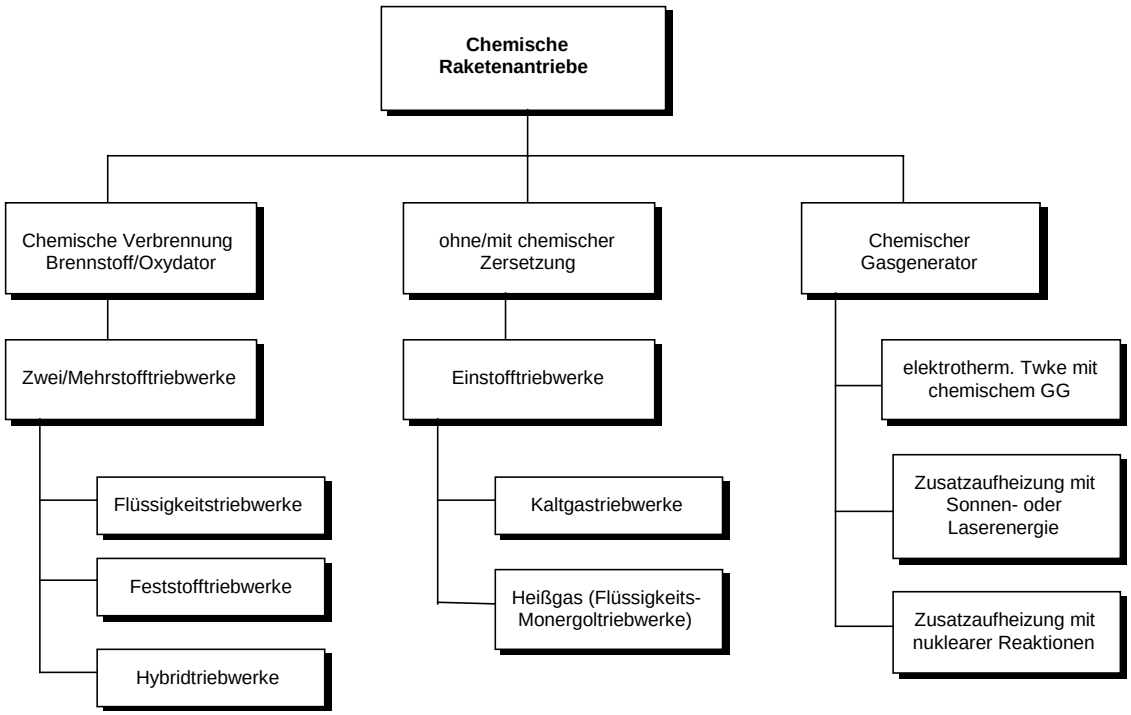


Abb. 5-1 Einteilung von Raumfahrtantrieben nach Art der Energieerzeugung im Arbeitsgas /27/

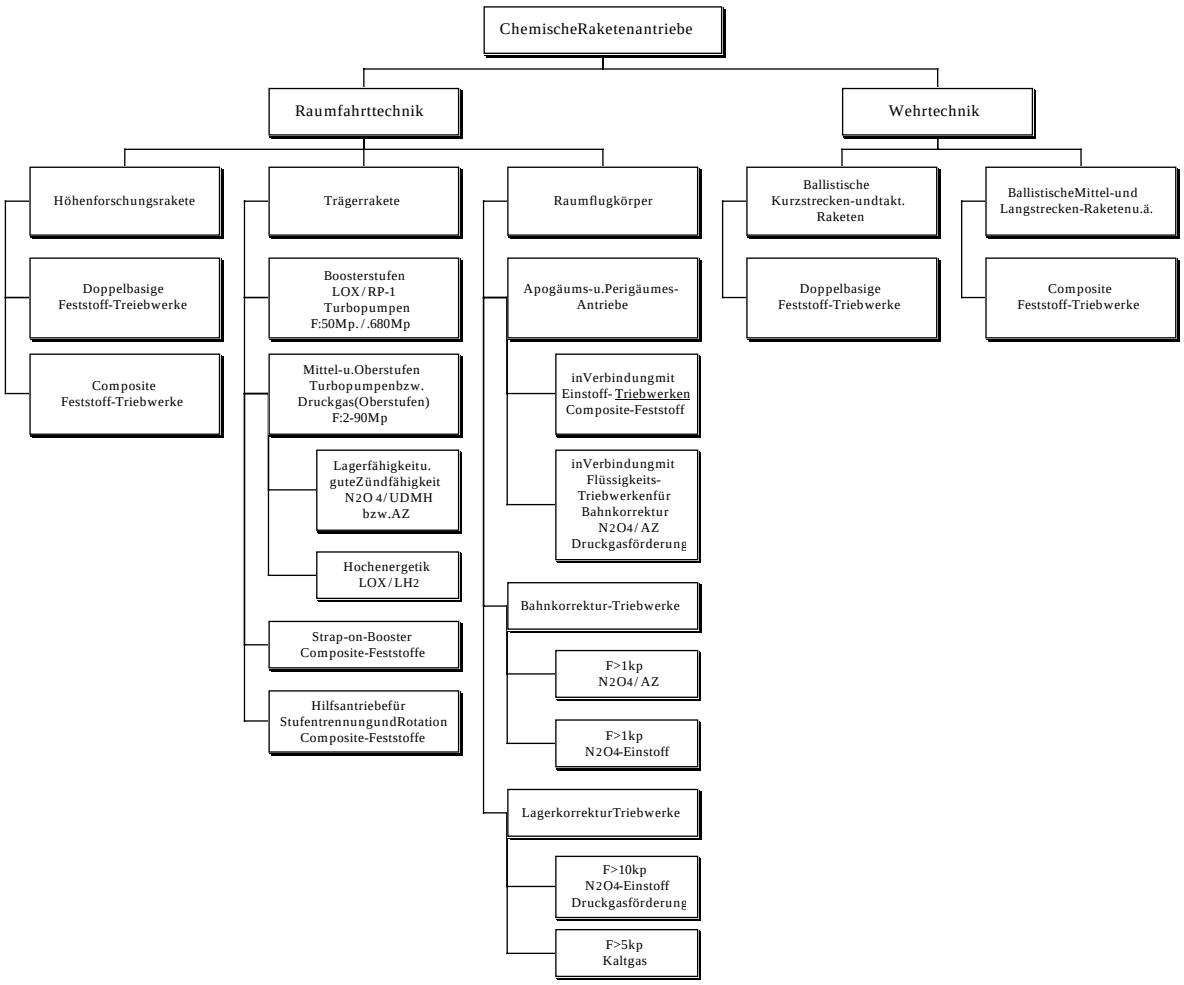


Abb. 5-2 Anwendung chemischer Raketenantriebe mit Angabe der derzeit bevorzugten Raketentreibstoffe, Treibstoffförderverfahren und Triebwerksschubklassen

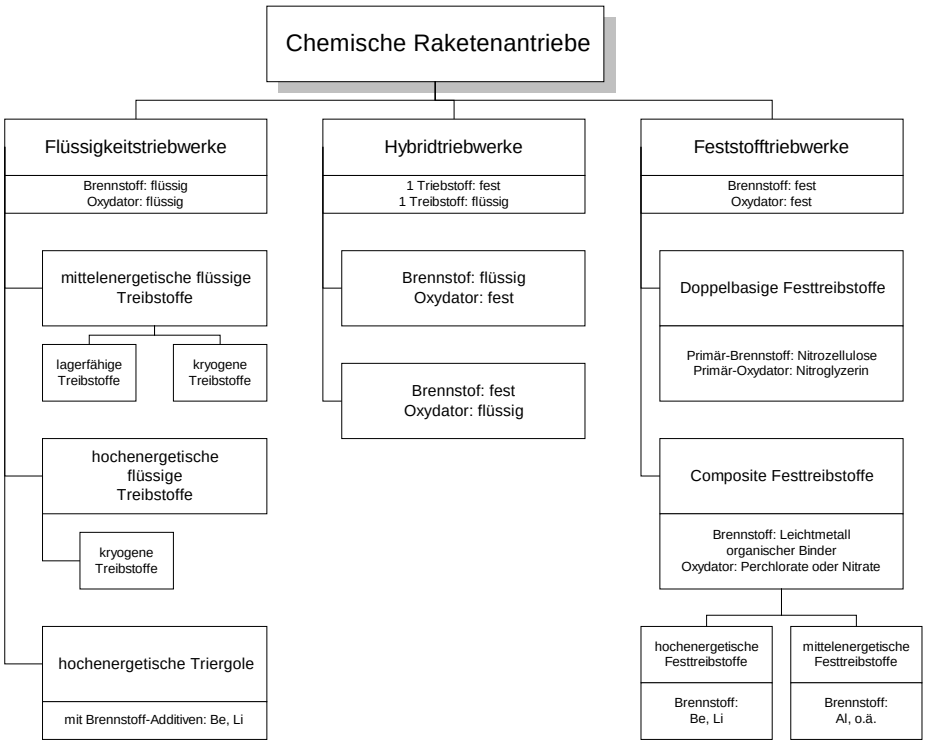


Abb. 5-3 Einteilung nach dem Aggregatzustand und der spezifischen Energie der Treibstoffe

5.1.2 Hierarchische Einteilungen

Die hierarchischen Einteilungen wären besonders geeignet für die Ermittlung der Wissenslücken, da die Objekte einer Hierarchieebene die Attribute der übergeordneten Ebene erben würden. Auf diese Weise ließen sich Attributlisten für ganze Objektklassen ermitteln oder im Bedarfsfall auch für ein bestimmtes Objekt. Hierzu müsste eine Einordnung auf einer möglichst tiefen Hierarchieebene erfolgen.

Dies setzt allerdings voraus, dass sich die in der Raumfahrt gebräuchlichen hierarchischen Einteilungen so verhalten wie semantische Netze oder Frames. Bei diesen in Kapitel 3.1 bereits beschriebenen Methoden der Wissensrepräsentation ist eine von der objektorientierten Programmierung bekannte Vererbung von Eigenschaften ein wichtiges Merkmal.

Dezimalklassifikation

In den 50er-Jahren wurde von E. Sänger der Versuch unternommen eine Dezimalklassifikation der Astronautik zu erstellen. Ihre Hauptgebiete sind nachfolgend aufgeführt. Die Raumtransportsysteme tauchen darin als eine Kategorie auf. Dagegen umfasst die Raketentechnik mehrere Kategorien. In Tab. 5-2 ist die erste und zweite Ebene dieser Klassifikation wiedergegeben. In Tab. 5-3 ist die Einteilung der Antriebe und in Tab. 5-4 die der Raumfahrzeuge gezeigt. Obwohl diese Klassifikation sehr detailliert und umfassend alle Gebiete der Raumfahrt (Astronautik) erfasst, ist sie für die hier gedachte Anwendung ungeeignet. Zwar kann man in der Einteilung der Antriebe durchaus einige Klassen identifizieren, aber es sind keinerlei Attribute mit den Klassen verbunden, so dass natürlich auch keine Vererbung stattfindet. Dagegen ist die Eignung als Zugriffshilfe und zum Auffinden bestimmter Informationen in der Datenbank unbestritten. Hierfür ist die Dezimalklassifikation aufgrund ihres Bekanntheitsgrades sogar noch besser geeignet als das AIM Schichtenmodell, welches in den nachfolgenden Abschnitten vorgestellt wird.

0.	Allgemeine Grundlagen	1.	Wissenschaftliche Grundlagen
00	Allgemeines	10	Allgemeines
01	Geschichte	11	Mathematik
02	Recht und Verschiedenes	12	Astronomie
03	Literatur und Philologie	13	Astrophysik
04	Politik und Wirtschaft	14	Geophysik und Geodäsie
05	Raumfahrtkosten	15	Geologie
06	Raumfahrtplanung	16	Geographie
07	Anwendung und Ergebnisse der Raumfahrt	17	Biologie
08	Auswirkungen	18	Biologische Medizin
09	Organisation	19	Menschliches Verhalten
2.	**Physikalische Grundlagen**	**3.**	**Chemische- und Technische Grundlagen**
20	Allgemeines	30	Allgemeines - Chemie
21	Theoretische Physik	31	Chemie
22	Relativistische Physik	32	Kosmochemie
23	Festkörper Physik	33	Treibstoffe
24	Thermodynamik	34	Allgemeines - Technologie
25	Thermodynamik und Strömungslehre	35	Baustoffe
26	Elektronik	36	Hilfsstoffe
27	Photonik	37	Umgebungseinfluss auf Werkstoffe
28	Atom und Molekularphysik	38	Herstellungsgrundlagen
29	Kern und Teilchenphysik	39	Maschinenelemente
4.	**Astrodynamik**	**5.**	**Astrionik**
40	Allgemeines	50	Allgemeines
41	Startverfahren	51	Messtechnik
42	Stabilität und Regelung	52	Regeltechnik
43	Aufstiegsbahnen	53	Rechengeräte und Automation
44	Freiflugbahnen	54	Photographie und Fernsehen
45	Satellitenmechanik	55	Energieerzeuger
46	Landeverfahren	56	Nachrichtentechnik
47	Flugbahnprofile	57	Komponenten der Navigationssysteme
48	Flugdatenauswertung	58	Komponenten der Regelsysteme
49	Relativistische Flugmechanik	59	Fernlenkmethoden und Systeme
6.	**Triebwerkssystem**	**7.**	**Luft- und Raumfahrzeuge**
60	Allgemeines	70	Allgemeines
61	Triebswerkstheorie	71	Entwurfsprinzipien
62	Triebswerkskomponenten	72	Statik und Festigkeitsrechnung
63	Triebwerkszubehör	73	Baugruppen
64	Luft- und Strahltriebwerke	74	Luftfahrzeuge
65	Feststoff und Hybrid Triebwerke	75	Flugkörper
66	Flüssigtreibstoff Raketentriebwerke	76	Raumfahrzeuge
67	Kernenergetische Triebwerke	77	Instrumentierte Satelliten
68	Elektrische Triebwerke	78	Unbemannte Raumfahrzeuge
69	Sondertriebwerke	79	Bemannte Raumfahrzeuge
8.	**Raumflugbetrieb**	**9.**	**Bodenanlagen und -ausrüstung**
80	Grundlagen	90	Bodenorganisation
81	Betriebsbedarf	91	Ausbildungseinrichtungen
82	Fahrzeugführung	92	Forschungseinrichtungen
83	Navigationsprinzipien	93	Entwicklungseinrichtungen
84	Zubehör	94	Produktionseinrichtungen
85	Flugausrüstung (beweglich)	95	Starteinrichtungen und Raumflughäfen
86	Startvorbereitungen	96	Bahnmess- und Funkeinrichtungen
87	Erd- Orbit Betrieb	97	Treibstoffproduktion und Versorgung
88	Erd- Lunar Betrieb	98	Transporteinrichtung
89	Erd- Planetar Betrieb	99	Sondergebiete

Tab. 5-2 Dezimalklassifikation der Raumfahrt (Hauptgruppen)

6	Antriebssysteme		
60	**Allgemeines**	**61**	**Triebwerkstheorie**
600	Allgemeines	610	Allgemeines
601	Antriebsmethoden	611	Leistungsanalyse
602	Klassifizierung der Antriebssysteme	612	Treibstoffförderung
603	Herstellungsprobleme	613	Treibstoffeinspritzung und Mischung
604	Prüfverfahren und -probleme	614	Verbrennungsparameter und Stabilität
605	Installierung der Antriebssysteme	615	Innenströmung
606	Zuverlässigkeit der Antriebssysteme	616	Wärmeübergang und Kühlung
607	Wartung und Reparatur im Betrieb	617	Außenströmung
608	Typensammlung der Antriebssysteme	618	Arbeitsprozesse
609	Sonstiges	619	Sonstiges
62	**Triebwerkskomponenten**	**63**	**Triebwerkszubehör**
620	Allgemeines	630	Allgemeines
621	Diffusoren	631	Rohrleitungen
622	Strömungsmaschinen	632	Anlasseinrichtungen
623	Einspritzsysteme	633	Zündanlagen
624	Verbrennungskammer	634	Überwachungssysteme
625	Ausströmdüse	635	Überwachungsgeräte
626	Gasgeneratoren	636	Regelsysteme und -geräte
627	Druckgasanlagen	637	Hydraulische Ausrüstung
628	Wärmetauscher, Kühler	638	Triebwerksaufhängung
629	Sonstiges	639	Sonstiges
64	**Luftatmende Triebwerke**	**65**	**Feststoff- und Hybridtriebwerke**
640	Allgemeines	650	Allgemeines
641	Kolbentriebwerke	651	Treibstoffladung
642	Turboprop Triebwerke	652	Gehäusekonstruktion und -herstellung
643	Plus- Triebwerke	653	Düsenkonstruktion und -herstellung
644	Turbo- Strahltriebwerke	654	Schubregelung
645	Turbo- Staustrahltriebwerke	655	Zünd- und Abschaltsysteme
646	Staustrahltriebwerke	656	Feststofftriebwerk (koml.)
647	Sondertriebwerke	657	GEL- Triebwerke
648	Andere kombinierte Triebwerke	658	Hybrid- Triebwerk
649	Sonstiges	659	Sonstiges
66	**Flüssigkeits- Raketentriebwerke**	**67**	**Kernenergetische Triebwerke**
660	Allgemeines	670	Allgemeines
661	Kleinsttriebwerke	671	Reaktortechnik
662	Monergol-Triebwerke	672	Strahlungsschutzanforderungen
663	Lagerfähige Triebwerke mit Druckgasförderung	673	Nukleare Flugzeugantriebe
664	Lagerfähige Triebwerke mit Pumpenförderung	674	Radioisotopen- Antriebe
665	Kryogene Triebwerke mit Druckgasförderung	675	Nukleare Festkernantriebe
666	Kryogene Triebwerke mit Pumpenförderung	676	Nukleare Gaskernantriebe
667	Mehrstofftriebwerke	677	Nukleare Pulsantriebe
668	Raketentriebwerke mit Luftzumischung	678	Fusionstriebwerke
669	Sonstiges	679	Sonstiges
68	**Elektrische Triebwerke**	**69**	**Sondertriebwerke**
680	Allgemeines	690	Allgemeines
681	Elektrothermische Antriebe	691	Heißwasserraketen
682	Thermo-Ionische Antriebe	692	Strahlungsdruck- Antriebe
683	Elektrostatische Antriebe	693	Freie-Radikale Antriebe
684	Elektromagnetische Antriebe	694	Solar-Thermische Antriebe
685		695	
686		696	
687		697	
688		698	Photonen-Antriebe
689	Sonstiges	699	Sonstiges

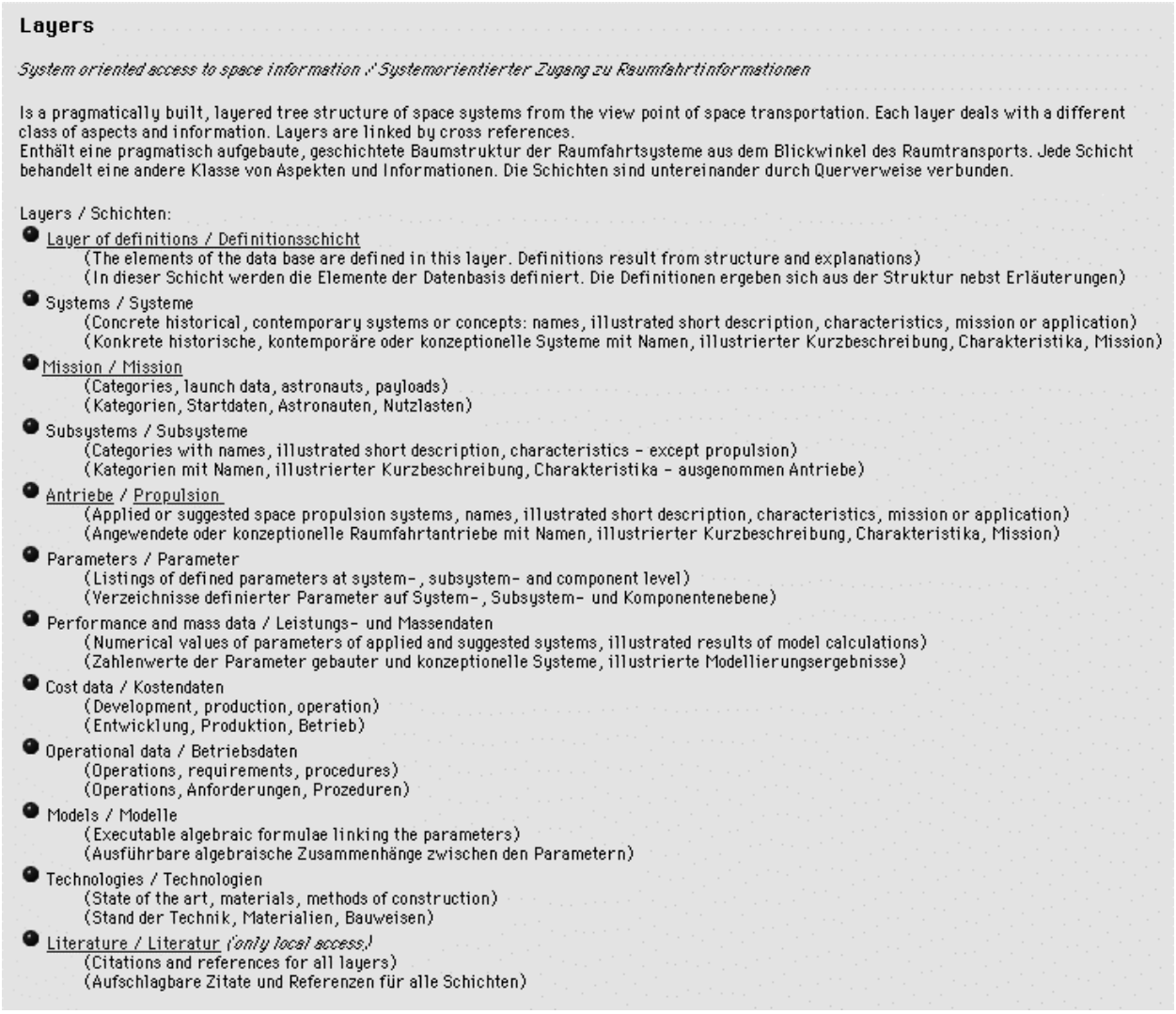
Tab. 5-3 Antriebssysteme in der Dezimalklassifikation der Raumfahrt

7	LUFT- UND RAUMFAHRZEUGE		
70	**Allgemeines**	**71**	**Entwurfsprinzipien**
700	Allgemeines	710	Allgemeines
701	Systemtechnik	711	Entwurfskriterien
702	Theorie der Raumfahrzeuge	712	Optimierungsverfahren
703	Versuche mit Raumfahrzeugen	713	Thermische Auslegung
704	Montage und Prüfmethoden	714	Auslegung der Zelle
705	Betriebsförderungen	715	Triebwerksauswahl
706	Zuverlässigkeitsbedingungen	716	Astrionische Systemauswahl
707	Nutzlast- Daten	717	Sicherheitsbedingungen
708	Typensammlungen	718	Bergungsmethoden
709	Sonstiges	719	Sonstiges
72	**Statik und Festigkeitsrechnung**	**73**	**Baugruppen**
720	Allgemeines	730	Allgemeines
721	Lastannahmen	731	Tragwerk
722	Rechenmethoden	732	Leitwerk
723	Festigkeitsrechnung	733	Zelle und Verkleidung
724	Belastungsversuche	734	Behälter und Leitungen
725	Aero- Elastizität	735	Schubgerüste
726	Dauerfestigkeit- und Vibrationsbeanspruchung	736	Landeausrüstung
727	Schwingungsberechnung	737	Bergungsausrüstung
728	Gewichts- und Momentenbestimmung	738	Kabine
729	Sonstiges	739	Sonstiges
74	**Luftfahrzeuge**	**75**	**Flugkörper**
740	Allgemeines	750	Allgemeines
741	Ballone und Luftschiffe	751	Kleine ballistische Flugkörper
742	Segel- und Gleitflugzeuge	752	Luft- Luft- Flugkörper
743	Luftkissenfahrzeuge	753	Luft- Boden- Flugkörper
744	Hubschrauber	754	Boden- Luft- Flugkörper
745	Propellerflugzeuge	755	Höhenraketen
746	Unterschallflugzeuge	756	Kleine Boden- Boden- Flugkörper
747	Überschallflugzeuge	757	Ballistische Flugkörper
748	Hyperschallflugzeuge	758	Ballistische Transport- Fahrzeuge
749	Sonstiges	759	Sonstiges
76	**Raumfahrzeuge**	**77**	**Instrumentierte Satelliten**
760	Allgemeines	770	Allgemeines
761	Trägerraketen (Verlustgeräte)	771	Entwurf, Entwicklung und Erprobung
762	Wiederverwendbare Trägerfahrzeuge	772	Geophysikalische Satelliten
763	Interplanetare Raumfahrzeuge	773	Meteorologische Satelliten
764	Raumfähren (Orbit- Orbit)	774	Navigations- und Geodätische Satelliten
765	Mond- Start- und Lande- Fahrzeuge	775	Nachrichten- Satelliten
766	Planetare Start- und Landefahrzeuge	776	Astronomische Satelliten
767	Planet- Erde- Wiedereintrittsfahrzeuge	777	Militärische Satelliten
768	Spezialraumfahrzeuge	778	Wissenschaftliche Instrumente in Satelliten
769	Sonstiges	779	Sonstiges
78	**Raumfluggeräte (unbemannt)**	**79**	**Bemannte Orbitalsysteme**
780	Allgemeines	790	Allgemeines
781	Entwurf, Entwicklung und Erprobung	791	Ballistische Kapseln
782	Mondfluggeräte	792	Wiedereintritts- Körper mit Auftrieb
783	Mond- Landegeräte	793	Raumlaboratorien
784	Planetarische Raumfluggeräte	794	Raumstationen
785	Planetarische Landegeräte	795	Bemannte militärische Satelliten
786	Interplanetarische Raumfluggeräte	796	Raumtaxis
787	Solare Raumfluggeräte	797	Werkzeugträger und Arbeitsgeräte
788	Wissenschaftliche Instrumente i. R.	798	Spezielle Hilfsausrüstung
789	Sonstiges	799	Sonstiges

Tab. 5-4 Dezimalklassifikation der Raumfahrt (Luft- und Raumfahrzeuge)

AIM Schichtenmodell

Das AIM Schichtenmodell wurde von Prof. Roger Lo für den Zugriff auf die Daten des Informationssystems und als pragmatische Strukturierung des Fachgebiets Raumfahrzeugtechnik vorgeschlagen und bereits teilweise in der WWW Zugriffskomponente realisiert. Es bot sich daher an, diese Einteilung auch für die Ermittlung der Wissenslücken einzusetzen. Die oberste Ebene des AIM Schichtenmodells, wie sie sich dem WWW Benutzer darbietet, ist in Abb. 5-4 gezeigt. Der Benutzer hat die Möglichkeit durch Hyperlinks in tiefere Ebenen des Schichtenmodells zu gelangen. Der Detailgrad steigt mit der Tiefe der Ebene. Das Schichtenmodell stellt eine nahezu ideale Annäherung an die Diskurswelt Raumfahrt dar. Durch die enge Verknüpfung der einzelnen Schichten, aber auch durch die Baumstruktur an sich wird ein sehr intuitiver Zugang zu den Informationen ermöglicht. Zur Zeit noch nicht realisiert, aber vorgesehen, ist auch ein Wechsel zwischen zwei Schichten innerhalb einer Ebene, so dass sich für den Benutzer eine dreidimensionale Baumstruktur mit Querverbindungen ergibt.



Layers

System oriented access to space information / Systemorientierter Zugang zu Raumfahrtinformationen

Is a pragmatically built, layered tree structure of space systems from the view point of space transportation. Each layer deals with a different class of aspects and information. Layers are linked by cross references.
Enthält eine pragmatisch aufgebaute, geschichtete Baumstruktur der Raumfahrtsysteme aus dem Blickwinkel des Raumtransports. Jede Schicht behandelt eine andere Klasse von Aspekten und Informationen. Die Schichten sind untereinander durch Querverweise verbunden.

Layers / Schichten:

- Layer of definitions / Definitionsschicht
  (The elements of the data base are defined in this layer. Definitions result from structure and explanations)
  (In dieser Schicht werden die Elemente der Datenbasis definiert. Die Definitionen ergeben sich aus der Struktur nebst Erläuterungen)
- Systems / Systeme
  (Concrete historical, contemporary systems or concepts: names, illustrated short description, characteristics, mission or application)
  (Konkrete historische, kontemporäre oder konzeptionelle Systeme mit Namen, illustrierter Kurzbeschreibung, Charakteristika, Mission)
- Mission / Mission
  (Categories, launch data, astronauts, payloads)
  (Kategorien, Startdaten, Astronauten, Nutzlasten)
- Subsystems / Subsysteme
  (Categories with names, illustrated short description, characteristics - except propulsion)
  (Kategorien mit Namen, illustrierter Kurzbeschreibung, Charakteristika - ausgenommen Antriebe)
- Antriebe / Propulsion
  (Applied or suggested space propulsion systems, names, illustrated short description, characteristics, mission or application)
  (Angewendete oder konzeptionelle Raumfahrtantriebe mit Namen, illustrierter Kurzbeschreibung, Charakteristika, Mission)
- Parameters / Parameter
  (Listings of defined parameters at system-, subsystem- and component level)
  (Verzeichnisse definierter Parameter auf System-, Subsystem- und Komponentenebene)
- Performance and mass data / Leistungs- und Massendaten
  (Numerical values of parameters of applied and suggested systems, illustrated results of model calculations)
  (Zahlenwerte der Parameter gebauter und konzeptionelle Systeme, illustrierte Modellierungsergebnisse)
- Cost data / Kostendaten
  (Development, production, operation)
  (Entwicklung, Produktion, Betrieb)
- Operational data / Betriebsdaten
  (Operations, requirements, procedures)
  (Operations, Anforderungen, Prozeduren)
- Models / Modelle
  (Executable algebraic formulae linking the parameters)
  (Ausführbare algebraische Zusammenhänge zwischen den Parametern)
- Technologies / Technologien
  (State of the art, materials, methods of construction)
  (Stand der Technik, Materialien, Bauweisen)
- Literature / Literatur *(only local access!)*
  (Citations and references for all layers)
  (Aufschlagbare Zitate und Referenzen für alle Schichten)

Abb. 5-4 Oberste Ebene des AIM Schichtenmodells

In Tab. 5-5 sind beispielhaft die weiteren Ebenen der Definitionsschicht gezeigt. Auch das AIM Schichtenmodell ist für die Bestimmung der Wissenslücken nicht geeignet. Die Klassifizierungstiefe ist nicht hoch genug und es sind ebenfalls keine Attribute zu den Klassen zugeordnet.

Raketen und Raumtransportsysteme	Raumtransportsysteme sind fast ausschließlich Anwendungen der Raketentechnik. Aus pragmatischen Gründen werden hier auch einige nicht dem Raumtransport dienende Raketen mit behandelt.
Erdgebundene Raketen	Erreichen keine orbitalen Geschwindigkeiten
Höhenraketen	für Meteorologie, Klimatologie
Militärische Flugkörper	Einteilung nach Reichweite
Taktische Raketen	Klassische Einteilung nach Lage von Abschussplatz und Ziel (Boden-Boden, Boden-Luft,...) oder Reichweite in Kurz- und Mittelstreckenraketen
Interkontinentale ballistische Raketen	Langstreckenraketen mit globaler Reichweite. Mit verringerter Nutzlast häufig orbitale Kapazität
Trägerraketen	fliegen vom Erdboden (SL: "Sea level") in den Weltraum
Expendable launch vehicles	Nicht-wiederverwendbar (ELVs: "Expendable Launchers")
Raumtransporter	sind per ad hoc Definition wiederverwendbar
Geflügelte Raumtransporter	Starten vertikal oder horizontal, landen immer aerodynamisch
Ballistische Raumtransporter	Starten und landen immer vertikal
Schwerlastraketen/Heavy Lift Launch Vehicles	können ganz oder teilweise wiederverwendbar sein.
Oberstufen, orbitale Fahrzeuge	dienen dem Transfer von einer Umlaufbahn zur anderen, vor allem im erdnahen und cislunaren Raum. Alle können expendable oder wiederverwendbar sein.
Spezielle Antriebsstufen	
orbitale Versorgungsstufen.	Beispiele: SOYUZ, Progress
Rückkehrfahrzeuge	haben nur Retroantriebe, nehmen also am Aufstiegsantrieb nicht teil. Wiederverwendbare fliegen als Nutzlasten nach LEO zurück. Beispiele: APOLLO Rückkehrkapseln und HERMES Konzept
Orbital Transfer Vehicles (OTVs)	Wiederverwendbare, raumstationierte Transferfahrzeuge im cislunaren Raum.
Raumfähren	Die eigentlichen OTVs mit großem ΔV
Raumschlepper	Orbiting Maneuvering Vehicles (OMVs) mit kleinem ΔV
Interplanetare Transferfahrzeuge	
Kickstufen und Sonden	sind unbemenscht. Beschleunigung (Kick) aus LEO nach GTO und GEO, oder von LEO in translunare oder transplanetare Bahnen. Dann typische Missionen: Vorbeiflüge oder Einfang in planetare Orbits mittels Retro-Antrieb.
Raumschiffe	transportieren neben Fracht auch Piloten oder Passagiere in translunare oder transplanetare Bahnen.
Interstellare Transferfahrzeuge	
Interstellare Sonden	Beispiele: Voyager, Daedalus.
Interstellare Raumschiffe	Vom Generationenflug bis zum relativistischen Flug
Extraterrestrische Landefahrzeuge	zum Transport zwischen Umlaufbahnen und Oberflächen von Himmelskörpern. Atmosphärensonden und Hartlander werden ebenso wie Oberflächenfahrzeuge bei Nutzlasten geführt.
Einweggeräte	
Reine Landefahrzeuge.	Beispiel: VIKING Marslander
Landefahrzeuge mit Rückkehrstufen.	Beispiel: APOLLO Mondfähren
Wiederverwendbare Landefahrzeuge	kehren mit Nutzlast in den Weltraum zurück. Mission analog jener irdischer Trägerraketen.
Mondshuttle:	Pendelverkehr zwischen LUO und Lunar Surface (auch: Lunar Bus)
Sonstige Landefahrzeuge:	Mars-Shuttle für Pendelverkehr zwischen LMO und Mars Surface
Bodeninfrastruktur	ist beschränkt auf terrestrische Anlagen für den Raumtransport
Raketenstartplätze	
Raumflughäfen	
Bodenstationen	

Orbitale Infrastruktur	(Die Trennung zwischen transportrelevanten Systemen und Nutzlasten ist hier vielfach schwierig und auch nicht sinnvoll. Hier: Verkehrsknoten, Kraftwerke, Forschungslabors, µ-g Produktionsanlagen, etc.)
Anlagen in GEO-Space	befinden sich in Erdumlaufbahnen (LEO, PEO, SSO, HEO bis GEO)
Raumstationen	sind bemenscht
LEO-Raumstationen.	Beispiele: MIR, ISS, Umladestationen, Umsteigestationen, Servicing Facilities, Weltraumhotels, Weltraumkliniken
GEO-Raumstationen	Beispiele: Satelliten Servicing Station, Solare Kraftwerke (SSPS), Orbital Tower
Satellitensysteme	sind unbemenscht oder werden höchstens von Astronauten gewartet
Telekommunikationssatelliten	(Geostationär; Molnya-Bahnen) Navigation, GPS, Meteorologie, Erdbeobachtungsplattformen, Krafft-Ehricke Strukturen)
Anlagen im cislunaren Raum	
LUO-Anlagen.	Beispiele: Lunar-LOX Depot; rotierende Astronauten-Rekonditionierungs-station.
Anlagen in Lagrange-Punkten im Erde-Mond System	Einschließlich der translunaren L3-Position (z.B.: L3-Halo-Datenrelais Satelliten).
Anlagen in Umlaufbahnen um andere Himmelskörper	
Anlagen in Umlaufbahnen um Planeten	Datenrelaissatelliten in Mars-Orbit
Anlagen in Umlaufbahnen um die Sonne	Beispiele: Cycling Spaceship Erde-Mars-Erde, interplanetarer Massenbeschleuniger.
Sonstige orbitale Infrastruktur	Daten- oder Laser Relaisstationen auf Solar Escape Bahnen.
Extraterrestrische Systeme	Die Abtrennung transportrelevanter Systeme von Nutzlasten ist auch hier vielfach schwierig und nicht sinnvoll.
Lunare Systeme , Mondstationen	Beispiele: Basis, Observatorium, Produktionsstätten, Kraftwerke
Startplätze	
Mondbodenverkehr	
Planetare Systeme	Einteilung wie A)
Sonstige Systeme	Beispiele: Weltraumkolonien, Kardashev Systeme
Nutzlasten / Missionen	
Forschung	
Im GEO-space	
Erdgerichtet	Geo-Wissenschaften, Klimatologie etc
In-situ	µ-g Plattformen, Extraterrestrik
Raumgerichtet	Hubble Space Telescope
Interplanetare Forschungsnutzlasten	
Atmosphärensonden	Beispiel: Galileo Jupiter Atmosphärensonde
Hartlander	Beispiel: Ranger Mondsonden
Oberflächenfahrzeuge	Lunar Rover
Anwendung	Einschließlich militärischer Anwendungen
Telekommunikation	Stationäre und mobile Kommunikation; Navigation
Microgravity	
Erdbeobachtung	Erderkundung, Wetter
Sonstige zivile Anwendungen	
Sonstige militärische Anwendungen	

Tab. 5-5 AIM Strukturierung des Sachgebietes

Anmerkungen zu hierarchischen Einteilungen

Es wurden mehrere Versuche unternommen für die wichtigsten Objekttypen der AIM Datenbank (`Launcher`, `Stage`, `Engine`) eine reine hierarchische Klassenstruktur ohne Mehrfachvererbung zu finden. Der Idealfall einer solchen Struktur ist in Abb. 5-5 Ausschnittsweise gezeigt. Hier ist eine normale Baumstruktur mit Vererbung dargestellt. Die Unterklassen erben die Attribute von den Oberklassen und somit ist eine eindeutige Zuordnung von Attributen zu bestimmten Knoten der Klassenhierarchie möglich. Je tiefer im Baum man die Zugehörigkeit bestimmen kann, um so mehr Attribute kann man eindeutig zuordnen. Wenn sich eine solche Struktur für die oben genannten Objekttypen finden ließe, müsste nur noch das Problem der Einordnung beliebiger Objekte in die Hierarchie gelöst werden. Dies würde entweder auf Basis der vorhandenen Attributmenge geschehen, wie in Kapitel 5.2 geschildert, oder über Attributwerte, die über die Zugehörigkeit zu einer bestimmten Klasse entscheiden.

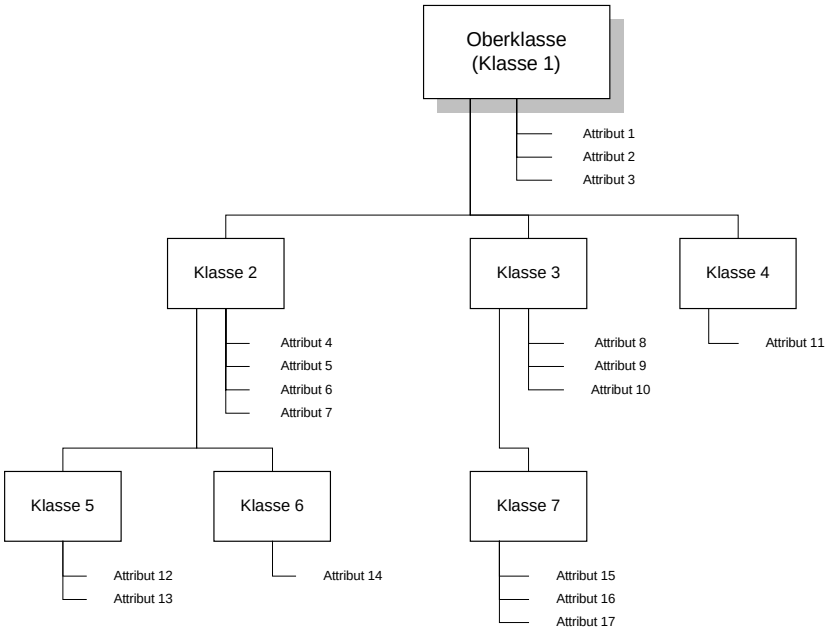


Abb. 5-5 Günstigster Fall: Klassenhierarchie ohne Mehrfachvererbung

Trotz intensivster Bemühungen konnte keine konsistente Struktur erzeugt werden. Es trat in nahezu allen Versuchen der Fall auf, dass eine Unterklasse zwar den Großteil der Attribute der Oberklasse erben konnte, aber ein oder zwei Attribute nicht zutrafen, so dass entweder mit Default oder Standardwerten gearbeitet werden musste oder man war gezwungen auf Mehrfachvererbung zurückzugreifen. Ein Beispiel für eine solche Hierarchie ist in Abb. 5-6 gezeigt.

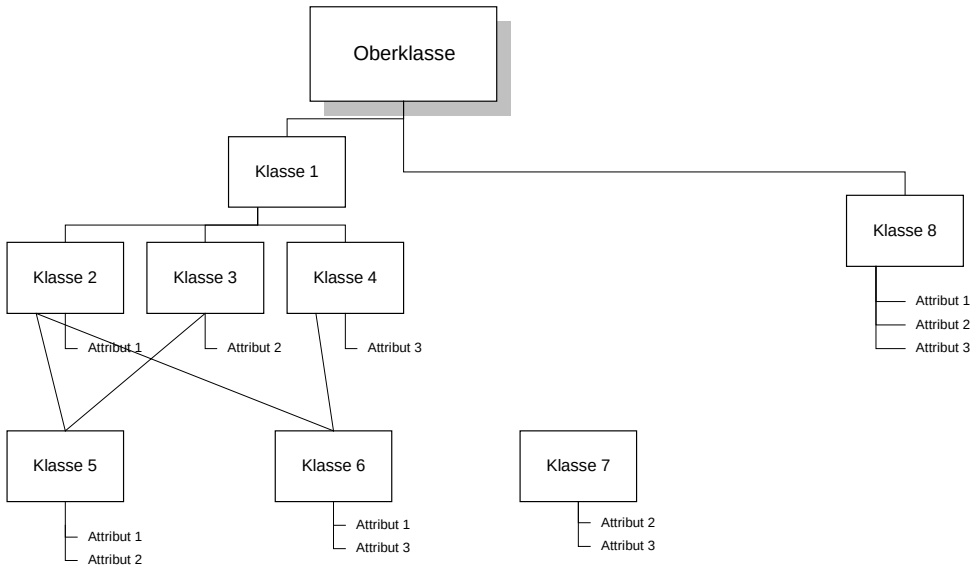


Abb. 5-6 Wahrscheinlichster Fall: Klassenhierarchie mit Mehrfachvererbung

Die Erzeugung einer Hierarchie quasi aus dem Nichts oder aufbauend auf bestehende Klassifikationen von Raumfahrtobjekten brachte keinen Erfolg. Entweder war man gezwungen auf Standardwerte und Mehrfachvererbung zurückzugreifen, um den Preis eines höheren Programmieraufwandes oder, wie im Fall der bestehenden Klassifikationen, war keine vollständige Erfassung aller Objekte in der notwendigen Tiefe möglich. Um dem entgegenzuwirken erschien es sinnvoll eine Struktur auf Basis der vorhandenen Daten zu erzeugen.

5.2 Objektklassen auf Basis von Attributgruppen

Der in den vorhergehenden Abschnitten beschriebene Weg, den Knoten bestehender Klassifikationen und Einteilungen Attributlisten zuzuordnen, stellte sich als sehr aufwendig heraus. Die dabei auftretende Hauptschwierigkeit ist die fehlende hierarchische Baumstruktur der meisten Klassifikationen und das mehrfache Auftreten von einzelnen Attributen an mehreren Knotenstellen. Um die aus der OOP (object oriented programming) bekannte Mehrfachvererbung (multiple inheritance) zu vermeiden, bietet es sich an auf Basis der bereits bestehenden Daten eine Klassifizierung abzuleiten.

Dabei wird davon ausgegangen, dass bei einem genügend großen Datenbestand (Auszug aus der Diskurswelt) für jede denkbare Klasse von Objekten mindestens ein Vertreter mit allen dazugehörigen Attributen bereits erfasst ist. Außerdem wird vorausgesetzt, dass der vorhandene Datenbestand frei von Fehlern ist. Insbesondere dürfen Objekte des Bestandes keine Attribute zugeordnet haben, die sie eigentlich nicht besitzen dürften. Da die in AIM vorhandenen Daten in der Regel aus der einschlägigen Literatur entnommen sind und eine fehlerhafte Eintragung relativ unwahrscheinlich ist, kann man davon ausgehen, dass der Datenbestand die wirklichen Objektklassen einigermaßen wiedergibt.

Allerdings wird keine Vollständigkeit der zu einer Klasse gehörenden Attribute vorliegen. Ganz im Gegenteil muss davon ausgegangen werden, dass der überwiegende Teil der gespeicherten Objekte noch unvollständige Attributmengen enthält. Gerade dies war ja der Grund für die Notwendigkeit der Klassifizierung, nämlich das Feststellen der fehlenden Attribute eines Objektes.

Für die Lösung dieses Problems bietet sich der Einsatz von Expertenwissen an. Wenn man zunächst einmal alle in der Datenbank vorhandenen Objektklassen ermittelt und übersichtlich darstellt, sollte es für einen Raumfahrtexperten möglich sein Klassen mit unsinnigen Attributkombinationen zu eliminieren und außerdem die Anzahl der gefundenen Klassen durch Attributabhängigkeiten zu reduzieren. Genau dies wurde für die drei wichtigsten Oberklassen (`Launcher`, `Stage`, `Engine`) der AIM Datenbank gemacht. Im Folgenden wird die dabei angewandte Vorgehensweise erläutert.

Ermittlung aller vorhandenen Attributkombinationen

Mit Hilfe eines Programms werden alle in der Datenbank vorhandenen, voneinander verschiedenen Attributkombinationen eines Objekttyps ermittelt und aufgelistet. Die Anzahl der so gefundenen Objektklassen kann mitunter sehr groß sein (z.B. `Engine`: 507 Klassen, 705 Objekte). Sie ist aber nicht unbedingt abhängig von der Anzahl der vorhandenen Objekte (z.B. `Launcher`: 40 Klassen, 307 Objekte).

Die Maximalanzahl möglicher Klassen wird durch die Attributanzahl nach folgender Formel aus der Kombinatorik bestimmt:

$$N_K = \binom{n_A}{0} + \binom{n_A}{1} + \binom{n_A}{2} + \cdots + \binom{n_A}{n_A} = 2^{n_A}$$

N_K : Anzahl verschiedener Objektklassen
n_A : Anzahl der Attribute

Die Anzahl verschiedener Objektklassen wird natürlich noch von der Anzahl der Objekte begrenzt. Es kann maximal so viele Klassen wie Objekte geben. In Abb. 5-7 ist der ungünstigste Fall für ein Beispiel mit 3 Attributen gezeigt. Obwohl eine Klasse mit Objekten völlig ohne Attribute keinen Sinn macht, ist sie der Vollständigkeit halber mit aufgeführt.

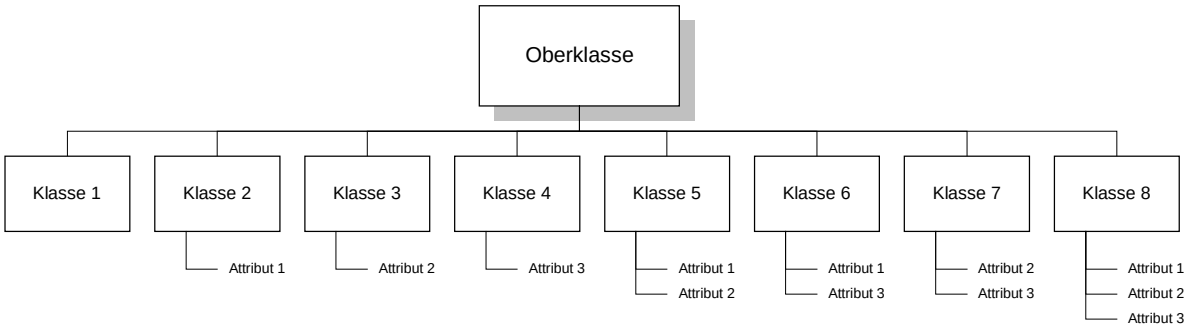


Abb. 5-7 Ungünstigster Fall: Alle möglichen Attributkombinationen kommen vor

Die so gefundenen Listen werden durch verschiedene Maßnahmen verkleinert. Attribute, die zu einem Subsystem gehören, werden zu logischen Gruppen zusammengefasst. Abhängigkeiten zwischen verschiedenen Attributen werden als Regeln formuliert und somit Existenzbedingungen für Attribute geschaffen. Als Ergebnis erhält man eine Anzahl von Attributmengen, die wiederum Klassen von Objekttypen beschreiben. Stellt man jetzt die Zugehörigkeit eines Objekts zu einer oder mehrerer dieser Klassen fest, kann man die Attributmenge, die dieses Objekt mindestens besitzen muss, anhand der Attributmengen der Klasse(n) feststellen.

Zusammenfassung von Attributen deren Existenz voneinander abhängt

Attribute die immer gemeinsam auftreten werden zu Attributgruppen zusammengefasst (z.B. `WENN Tank Diameter DANN Tank Mass UND Tank Volume`). So kann bei Vorhandensein eines Attributs aus der Attributgruppe auf die Notwendigkeit des Vorhandenseins aller anderen Attribute geschlossen werden. Außerdem verringert sich die Gesamtzahl der Attribute durch Bildung von Attributgruppen erheblich. Bei der Zusammenfassung von Attributen die technische Eigenschaften beschreiben (im Gegensatz zu nicht-technischen Attributen wie Datumsangaben, Organisationen u.ä.) zu Attributgruppen, zeigt sich, dass dieses Zusammenfassen letztendlich auf das Vorhandensein oder nicht Vorhandensein von Subsystemen und Komponenten hinausläuft. Einige der Komponenten sind dann voneinander abhängig oder schließen einander aus, was die Kombinationen von Attributen zu Kombinationen von Subsystemen werden lässt. Dies wiederum führt dann zu Regeln für die Existenz von Attributen.

Aufstellen von Regeln für die Existenz von Attributen

Es werden Regeln für das Vorhandensein von Attributen in Abhängigkeit von der Existenz anderer Attribute aufgestellt (beispielsweise `WENN Development End DANN Development Start`). Bei einigen Attributen können durch ihr Vorhandensein Rückschlüsse auf eine ganze Reihe anderer Attribute geschlossen werden. Bei den Datumsangaben etwa handelt es sich hauptsächlich um Schritte im Laufe des Entwicklungsprozesses, die in immer derselben Reihenfolge auftauchen. Das heißt, dass die Existenz eines Entwicklungsschritts automatisch die Existenz der vorhergehenden Schritte bedingt. Somit ergeben sich bei den Datumsangaben aus der Reihenfolge automatisch Existenzbedingungen für die vorhergehenden Schritte.

Außerdem können Attribute definiert werden, deren Vorhandensein die Existenz von Subsystemen und Komponenten bedingt. Die Existenz der mit diesen Subsystemen und Komponenten verbundenen Attribute oder Attributgruppen kann dann natürlich ebenfalls gefolgert werden (z.B. `WENN Oxidizer Tank Diameter DANN` Komponente `Oxidizer Tank` vorhanden `DANN` alle Attribute der Komponente `Oxidizer Tank` vorhanden). Zusätzlich kann das Vorhandensein eines Subsystems oder einer Komponente das Vorhandensein eines weiteren Subsystems oder Komponente bedingen. Auch dies kann als Regel ausgedrückt werden. Durch die Anwendung der Regeln auf die im ersten Schritt gefundenen Klassen reduziert sich deren Anzahl zum Teil erheblich.

5.3 Objektklassen in AIM

Für die wichtigsten Objekte in der AIM Datenbank wurden mit der in Abschnitt 5.2 beschriebenen Vorgehensweise Klassenlisten erzeugt. Hier soll beschrieben werden, welche Probleme auftraten und für welche Objekte dies durchgeführt wurde. Die Klassifizierung und deren Ergebnisse werden beschrieben, die gefundenen Klassenstrukturen kurz vorgestellt und das Ergebnis interpretiert. Die Verwendung der Klassenstrukturen für das Auffinden von Wissenslücken und dabei auftretende Probleme werden erläutert.

5.3.1 Klassifizierung durch Attributgruppen

Für die Objekttypen `Launcher`, `Stage` und `Engine` wurde je ein HyperCard-Stapel geschrieben, der für alle Objekte des jeweiligen Objekttyps die vorhandenen Attributkombinationen aus der Datenbank abfragt und eine Liste erstellt, die alle verschiedenen Kombinationen enthält. Dann werden diese Listen sortiert und eine tabellarische Darstellung erzeugt, die die verschiedenen Kombinationen erfasst. Diese Klassentabelle kann gespeichert und kann dann in ein Tabellenkalkulationsprogramm (hier MS Excel) importiert werden, wo sie weiterbearbeitet wird. Eine solche importierte Tabelle ist in Abb. 5-8 gezeigt.

	Klasse 1	Klasse 2	Klasse 3	Klasse 4	…	Klasse n
Attribut 1	x	x		x		x
Attribut 2		x	x			x
Attribut 3	x			x		
Attribut 4		x	x	x		
⋮						⋮
Attribut n	x		x		…	x

Abb. 5-8 Beispiel für eine importierte Klassentabelle

In der Tabellenkalkulation sind sowohl die Gruppen, die die einzelnen Subsysteme repräsentieren gespeichert, als auch die Abhängigkeitsregeln tabellarisch erfasst. Die Gruppen werden durch eine zweispaltige Tabelle erfasst, wie sie in Abb. 5-9 exemplarisch dargestellt ist. Die erste Spalte enthält die Gruppenbezeichnung, in der Regel der Name des Subsystems. Die zweite Spalte enthält alle Attribute, die immer zu dieser Gruppe gehören. Wenn die Attributmenge einer Klasse Gruppen enthält, dann empfiehlt es sich aus Gründen der Übersichtlichkeit stellvertretend für alle Attribute einer Gruppe den Gruppennamen zu verwenden. Dadurch wird die Anzahl der in der Klassentabelle darzustellenden Attribute teilweise stark reduziert, was den Überblick über die Klassen erleichtert.

Subsystem 1	Attribut 1
	Attribut 3
	Attribut 5
Subsystem 2	Attribut 4
	Attribut 9
	Attribut 11
⋮	⋮
Subsystem n	Attribut 8
	Attribut 15

Abb. 5-9 Beispiel für die Gruppenbildung durch Subsysteme

Die Abhängigkeitsregeln werden in einer Kreuztabelle eingetragen. Die erste Zeile und erste Spalte enthalten die Attribute in derselben Reihenfolge. Das betreffende Attribut steht in der ersten Spalte. Alle Attribute, die bei Vorhandensein dieses Attributs ebenfalls vorhanden sein müssen, erhalten ein Kreuz in der Spalte, die sich unterhalb ihres Eintrags in der ersten Zeile befindet. In der Abb. 5-10 sind beispielsweise folgende Regeln repräsentiert:

```
WENN Gruppe 1 DANN Gruppe 2
WENN Attribut 1 DANN Attribut 2
WENN Attribut 2 DANN Gruppe 1
WENN Attribut 2 DANN Attribut 1
WENN Attribut 2 DANN Attribut 3
WENN Attribut 3 DANN Gruppe 2
WENN Attribut 3 DANN Attribut 2
```

Eine Spalte die komplett mit Kreuzen gefüllt ist, kennzeichnet ein Attribut, dass in jedem Fall vorhanden sein muss. Durch das komplette Ausfüllen einer Spalte wird erreicht, dass bei Vorhandensein irgendeines anderen Attributs, die Bedingung für das Attribut welches in jedem Fall vorhanden sein soll, immer erfüllt ist. Existieren mehrere solcher Attribute, empfiehlt es sich eine Gruppe einzuführen, bei der dann die komplette Spalte ausgefüllt wird. In der Attributspalte bzw. Attributzeile müssen bei Vorhandensein von Gruppen natürlich an Stelle der Attribute die Attributgruppen oder Subsysteme stehen. Da jedes Attribut aus einer Gruppe quasi stellvertretend für die gesamte Gruppe steht, werden durch diese Maßnahme nicht mehr Abhängigkeiten von Attributen, sondern von Subsystemen definiert. Ein Nachteil der tabellarischen Darstellung ist der, dass kombinierte Abhängigkeiten nicht erfasst werden können. Eine Regel der Form

```
WENN Attribut 4 UND Attribut 5 DANN Attribut 7
```

kann beispielsweise nicht dargestellt werden. Dies ist jedoch nur ein geringfügiger Nachteil, da Regeln dieses Typs nur selten vorkommen. Bei der Definition der für die Klassen `launcher`, `stage` und `engine` notwendigen Regeln tauchte dieser Fall beispielsweise nur bei der Klasse `engine` einmal auf.

	Gruppe 1	**Gruppe 2**	…	**Gruppe n**	Attribut 1	Attribut 2	Attribut 3	…	Attribut n
Gruppe 1		x							
Gruppe 2									
⋮									
Gruppe n									
Attribut 1						x			
Attribut 2	x				x		x		
Attribut 3		x				x			
⋮									⋮
Attribut n							x	…	

Abb. 5-10 Speicherung der Abhängigkeitsregeln

Ein VisualBasic Programm übernimmt dann die Auswertung und Anwendung der Regeln auf die Klassentabelle. Alle Attribute und Subsysteme werden in einer Schleife überprüft und die abhängigen Attribute und Subsysteme erhalten ein Kreuz in der Klassentabelle. Dadurch wird die Anzahl der Klassen reduziert, da nun unter Umständen Klassen mit gleichen Attributlisten entstehen. Im nächsten Schritt werden alle doppelten Klassen entfernt und übrig bleibt die endgültige Klassenliste. Zu beachten

ist dabei, dass dieses Programm keine Kettenabhängigkeiten berücksichtigt. Wenn z.B. folgende Regeln definiert wären:

```
WENN Attribut 1 DANN Attribut 5
WENN Attribut 2 DANN Attribut 1
WENN Attribut 4 DANN Attribut 2
```

und in der Klassentabelle in einer Klasse nur das Attribut 4 vorhanden ist, dann würde nach Ablauf des Programms für diese Klasse nur die unterste Regel berücksichtigt werden. Da die Attribute sequentiell abgearbeitet werden, würde im vierten Durchlauf wegen der untersten Regel, ein Eintrag bei Attribut 2 vorgenommen werden. Da Attribut 2 jedoch schon behandelt worden ist, kommt bei dieser Klasse die zweite Regel nicht zur Anwendung. Dasselbe gilt für die oberste Regel. Führt man das Programm jedoch erneut aus, befindet sich nun in der erwähnten Klasse bei Attribut 4 und bei Attribut 2 ein Kreuz. Dadurch würde diesmal die zweite Regel angewendet werden, jedoch aus den genannten Gründen die dritte Regel nicht. In dem gezeigten Beispiel müsste das Programm dreimal aufgerufen werden, damit alle drei Regeln berücksichtigt werden.

Ein Hauptproblem bei dieser Vorgehensweise besteht auch darin, dass, wenn neue Attribute eingetragen oder nicht mehr benötigte Attribute gelöscht werden, sich die Anzahl möglicher Klassen verändert. Durch Änderung bei den Attributen eines Objekts (ein bereits vorhandenes Attribut oder ein völlig neues, bisher nicht gespeichertes Attribut, wird zu diesem Objekt hinzugefügt) kann eine Klasse wegfallen, da die Attributliste nun identisch mit einer anderen Klasse ist. Es kann auch eine neue Klasse entstehen, da die nun vorhandene Attributliste noch nicht erfasst wurde. Da ständig neue Einträge in der Datenbank vorgenommen werden, muss der in den vorhergehenden Abschnitten beschriebene Vorgang in regelmäßigen Abständen wiederholt werden, um die aktuelle Klassenstruktur zu erfassen. Wird dies nicht getan, arbeitet das Expertensystem normal weiter, allerdings besteht die Gefahr, dass nicht alle fehlenden Attribute eines Objekts richtig erkannt werden.

Ein weiteres Problem wurde bereits angesprochen. Die hier beschriebene Vorgehensweise geht davon aus, dass in der Datenbank keine falschen Attributzuordnungen vorgenommen wurden. Beim Löschen nicht benötigter Attribute sollte man sich genau überlegen, ob man das Attribut nicht zwecks besserer Klassenzuordnung in der Datenbank belässt. Daher sollten Änderungen an den verwendeten Attributen (löschen, Bezeichnung ändern, etc.) nur von Experten vorgenommen werden. Die Zuordnung von bestehenden Attributen, d.h. Eintragen neuer Daten, kann dagegen (ohne größere Überlegungen) von jedermann durchgeführt werden. Allerdings sollte man sich absolut sicher sein, dass der einzutragende Wert, d.h. das Attribut, auch wirklich zu dem entsprechenden Objekt gehört.

In den folgenden Abschnitten sind die entsprechenden Übersichtstabellen und Klassenlisten für die Objektklassen `Launcher`, `Stage` und `Engine` aufgeführt und kurz erläutert.

Launcher (Trägersystem)

Bei der Klasse `launcher` war es nicht möglich Attributgruppen zu finden. Dies liegt hauptsächlich daran, dass sämtliche Subsysteme dieser Klasse bereits als Entity-Tabellen vorliegen. Die Durchsuchung der Datenbank ergab 45 unterschiedliche Attributkombinationen. Diese konnten mit Hilfe der aus Tab. 5-6 ersichtlichen Abhängigkeitsregeln auf 25 Klassen reduziert werden, deren Attributkombinationen aus Tab. 5-7 ersichtlich sind. Die Abhängigkeitsregeln beim Objekttyp `launcher` beschränken sich nahezu auf die Regeln für die Datumseinträge. Diese geben den Lebenszyklus eines Trägersystems wieder. Hier bedingt jedes nachfolgende Ereignis alle Vorhergehenden. Die sich ergebende Klassenliste bleibt noch interpretationsfähig. Da sich die Klassenzugehörigkeit allein auf die in der Datenbank vorhandenen Attributkombinationen stützt, kann davon ausgegangen werden, dass noch zu viele Klassen vorhanden sind. Andererseits können noch nicht vorhandene Attributkombinationen per Hand hinzugefügt werden, wenn eine entsprechende Klasse von Trägern in der realen Welt ausgemacht werden kann.

	Conceptual Design Start	Conceptual Design End	Prelimenary Design Start	Prelimenary Design End	Program/Project Definition Start	Program/Project Definition End	Development Start	Development End	Final Design Review	Production Start	First Flight Attempt	Succesful Flight Demonstration	Production End	Last Flight	Final Documentation/Report	Abolishment Cost	Maintenance And Refurbishment Cost	Indirect Operation Cost	Development Cost	Direct Operation Cost	Launch Cost	Launch Site And Range Cost	Life Cycle Cost	Manufacturing Cost	Operation Cost	Vehicle Amortization Cost
Conceptual Design Start																										
Conceptual Design End	x																									
Prelimenary Design Start	x	x																								
Prelimenary Design End	x	x	x																							
Program/Project Definition Start	x	x	x	x																						
Program/Project Definition End	x	x	x	x	x																					
Development Start	x	x	x	x	x	x													x							
Development End	x	x	x	x	x	x	x																			
Final Design Review	x	x	x	x	x	x	x	x																		
Production Start	x	x	x	x	x	x	x	x	x															x		
First Flight Attempt	x	x	x	x	x	x	x	x	x	x										x						
Succesful Flight Demonstration	x	x	x	x	x	x	x	x	x	x	x															
Production End	x	x	x	x	x	x	x	x	x	x	x	x														
Last Flight	x	x	x	x	x	x	x	x	x	x	x	x	x													
Final Documentation/Report	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x										
Abolishment Cost																			x				x	x		
Maintenance And Refurbishment Cost																										
Indirect Operation Cost																										
Development Cost																										
Direct Operation Cost																										
Launch Cost												x														
Launch Site And Range Cost																										
Life Cycle Cost																										
Manufacturing Cost										x									x							
Operation Cost																					x			x		
Vehicle Amortization Cost																										
Average Launch Rate																										
Baufaktor																										
Deltavgeo																										
Deltavleo																										
Groß Factor																										
Growth Factor																										
Overall Mass Ratio																										
Payload Ratio																										
Propellant Ratio																										
Structure Ratio																										
Main Designer																										
Developer																										
Main Developer																										
Development Manager																										
Launch Service Agency																										
Manufacturer																										
Customer																										
Vehicle Operator																										
Dry Mass																										
Lift-Off Mass																										
Net Mass																										
Propellant Mass																										
Usable Propellant Mass																										
Maximum Diameter																										
Overall Length																										

Tab. 5-6a Abhängigkeitsregeln der Oberklasse Launcher

	Average Launch Rate	Baufaktor	Deltavgeo	Deltavleo	Groß Factor	Growth Factor	Overall Mass Ratio	Payload Ratio	Propellant Ratio	Structure Ratio	Main Designer	Developer	Main Developer	Development Manager	Launch Service Agency	Manufacturer	Customer	Vehicle Operator	Dry Mass	Lift-Off Mass	Net Mass	Propellant Mass	Usable Propellant Mass	Maximum Diameter	Overall Length
Conceptual Design Start											x									x				x	x
Conceptual Design End																				x				x	x
Prelimenary Design Start																				x				x	x
Prelimenary Design End																				x				x	x
Program/Project Definition Start																				x				x	x
Program/Project Definition End																				x				x	x
Development Start												x								x				x	x
Development End																				x				x	x
Final Design Review																				x				x	x
Production Start																x				x				x	x
First Flight Attempt																				x				x	x
Succesful Flight Demonstration	x																			x				x	x
Production End																				x				x	x
Last Flight															x					x				x	x
Final Documentation/Report																				x				x	x
Abolishment Cost																				x				x	x
Maintenance And Refurbishment Cost																				x				x	x
Indirect Operation Cost																				x				x	x
Development Cost												x								x				x	x
Direct Operation Cost																		x		x				x	x
Launch Cost	x										x	x		x			x	x		x				x	x
Launch Site And Range Cost																				x				x	x
Life Cycle Cost																				x				x	x
Manufacturing Cost																x				x				x	x
Operation Cost															x		x	x		x				x	x
Vehicle Amortization Cost																				x				x	x
Average Launch Rate											x	x		x			x	x		x				x	x
Baufaktor																				x	x		x	x	x
Deltavgeo																				x		x		x	x
Deltavleo																				x		x		x	x
Groß Factor																				x				x	x
Growth Factor																			x	x				x	x
Overall Mass Ratio																			x	x	x			x	x
Payload Ratio																				x				x	x
Propellant Ratio																				x			x	x	x
Structure Ratio																			x	x				x	x
Main Designer																				x				x	x
Developer																				x				x	x
Main Developer												x								x				x	x
Development Manager																				x				x	x
Launch Service Agency												x					x	x		x				x	x
Manufacturer																				x				x	x
Customer												x					x	x		x				x	x
Vehicle Operator												x					x			x				x	x
Dry Mass																				x	x	x		x	x
Lift-Off Mass																				x				x	x
Net Mass																			x	x		x		x	x
Propellant Mass																			x	x	x		x	x	x
Usable Propellant Mass																			x	x		x		x	x
Maximum Diameter																				x				x	x
Overall Length																				x				x	x

Tab. 5-6b Abhängigkeitsregeln der Oberklasse Launcher

Attribut	Klassen																								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Conceptual Design Start																						x	x	x	x
Conceptual Design End																						x	x	x	x
Prelimenary Design Start																						x	x	x	x
Prelimenary Design End																						x	x	x	x
Program/Project Definition Start																						x	x	x	x
Program/Project Definition End																						x	x	x	x
Development Start																						x	x	x	x
Development End																									x
Final Design Review																									x
Production Start																									x
First Flight Attempt																									x
Succesful Flight Demonstration																			x	x	x		x	x	x
Production End																									x
Last Flight																									x
Final Documentation/Report																									x
Abolishment Cost																									x
Maintenance And Refurbishment Cost																									x
Indirect Operation Cost																									x
Development Cost																						x	x	x	x
Direct Operation Cost																									x
Launch Cost																			x	x	x		x	x	x
Launch Site And Range Cost																									x
Life Cycle Cost																									x
Manufacturing Cost																									x
Operation Cost																									
Vehicle Amortization Cost																									x
Average Launch Rate																		x	x	x	x		x	x	x
Baufaktor																									x
Deltavgeo																									x
Deltavleo																									x
Groß Factor																									x
Growth Factor																									x
Overall Mass Ratio																									x
Payload Ratio																									x
Propellant Ratio																									x
Structure Ratio																									x
Main Designer															x	x	x	x	x	x	x	x	x	x	x
Developer					x	x	x	x	x	x	x	x	x	x			x	x	x	x	x	x	x	x	x
Main Developer											x	x	x	x											
Development Manager				x						x								x	x	x	x		x	x	x
Launch Service Agency								x	x								x	x			x			x	x
Manufacturer			x	x						x			x	x		x	x	x			x	x	x	x	x
Customer							x	x	x	x		x		x			x	x	x	x	x		x	x	x
Vehicle Operator							x	x	x	x		x		x			x	x	x	x	x		x	x	x
Dry Mass		x				x			x		x									x					x
Lift-Off Mass	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Net Mass		x				x			x		x									x					x
Propellant Mass		x				x			x		x									x					x
Usable Propellant Mass		x				x			x		x									x					x
Maximum Diameter	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Overall Length	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Tab. 5-7 Weitere Unterteilung der Oberklasse `Launcher` (nach Anwendung der Regeln)

Stage (Stufe)

Da als Subsysteme einer Stufe bisher nur die Triebwerke als Entity-Tabelle ausgegliedert wurden, lassen sich verhältnismäßig viele Attributgruppen definieren. Man erkennt an Tab. 5-8, dass sich aus den dort aufgelisteten Gruppen die Entity-Tabellen `Tank` und `Propellant` erzeugen lassen. Da `Propellant` schon existiert würde es sich anbieten eine Tabelle `Stage_Propellant` zu erzeugen, in der festgelegt würde, um welchen Treibstoff es sich handelt. Da der Treibstoff aber durch die Triebwerke bestimmt wird und nicht von der Stufe allein abhängt, würde man in `Stage_Propellant` nur die Attribute speichern.

Bei `Tank` dagegen handelt es sich um ein echtes Subsystem, welches nochmals in Unterklassen (Fuel Tank, Oxidizer Tank, Other Tank) unterteilt werden kann. Aus Sicht der Datenmodellierung würde eine Ausgliederung des Objekttyps Tank aber nur Sinn machen, wenn ein Tank in mehreren Stufen Verwendung findet. Die restlichen Gruppen sind eigentlich Subsysteme von `Launcher`, wurden aber der Einfachheit halber in die Tabelle `Stage` integriert.

Aus den 66 Klassen vor der Zusammenfassung der Gruppen und der Anwendung der Regeln wurden die in Tab. 5-10 gezeigten 24 Klassen des Objekttyps `Stage`.

Fuel Tank	Fuel Tank Amount	**Propellant**	Propellant Ratio
	Fuel Tank Diameter		Propellant Residuals Mass
	Fuel Tank Length		Propellant Total Mass
	Fuel Tank Pressure		Usable Extra Propellant Mass
	Fuel Tank Volume		Usable Propellant Mass
	Propellant Fuel Mass	**Propellant Other**	Propellant Other Tank Amount
Oxidizer Tank	Oxidizer Tank Amount		Propellant Other Tank Diameter
	Oxidizer Tank Diameter		Propellant Other Tank Length
	Oxidizer Tank Length		Propellant Other Tank Pressure
	Oxidizer Tank Pressure		Propellant Other Tank Volume
	Oxidizer Tank Volume		Propellant Other Mass
	Propellant Oxidizer Mass	**PL Interface**	PL Interface Diameter
Tank	Tank Amount		PL Interface Mass
	Tank Diameter		PL Interface Length
	Tank Length	**Equipment Bay**	Equipment Bay Diameter
	Tank Pressure		Equipment Bay Length
	Tank Volume		Equipment Bay Mass
Interstage Adapter	Interstage Adapter Mass		VEB Interface Diameter
	Interstage Adapter Diameter	**Pressurization Gas**	Pressurization Gas Mass
	Interstage Adapter Length		Pressurization Gas Tank Pressure

Tab. 5-8 Attributgruppen der Oberklasse `Stage`

	Pressurization Gas	**Propellant**	**Tank**	**Propellant Other**	**Oxidizer Tank**	**Fuel Tank**	**Equipment Bay**	**PL Interface**	**Interstage Adapter**	Conceptual Design Start	Conceptual Design End	Prelimenary Design Start	Prelimenary Design End	Program/Project Definition Start	Program/Project Definition End	Development Start	Development End	Final Design Review	First Ignition Test	Final Burn Time Test	Production Start	First Flight Attempt	Succesful Flight Demonstration	Production End	Last Flight	Final Documentation/Report
Pressurization Gas		x	x																							
Propellant			x																							
Tank		x																								
Propellant Other		x				x																				
Oxidizer Tank		x				x																				
Fuel Tank		x																								
Equipment Bay																										
PL Interface																										
Interstage Adapter																										
Conceptual Design Start																										
Conceptual Design End										x																
Prelimenary Design Start										x	x															
Prelimenary Design End										x	x	x														
Program/Project Definition Start										x	x	x	x													
Program/Project Definition End										x	x	x	x	x												
Development Start										x	x	x	x	x	x											
Development End										x	x	x	x	x	x	x										
Final Design Review										x	x	x	x	x	x	x	x									
First Ignition Test										x	x	x	x	x	x	x	x	x								
Final Burn Time Test										x	x	x	x	x	x	x	x	x	x							
Production Start										x	x	x	x	x	x	x	x	x	x	x						
First Flight Attempt										x	x	x	x	x	x	x	x	x	x	x	x					
Succesful Flight Demonstration										x	x	x	x	x	x	x	x	x	x	x	x	x				
Production End										x	x	x	x	x	x	x	x	x	x	x	x	x	x			
Last Flight										x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
Final Documentation/Report										x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
Baufaktor		x																								
Growth Factor																										
Overall Mass Ratio		x																								
Structure Ratio																										
Actuator System Mass																										
Burn-Out Mass																										
Dry Mass																										
Groß Mass																										
Guidance and Control Mass																										
Lift-Off Mass																										
Net Mass																										
Propulsion System Mass																										
Recovery System Mass																										
Structure Mass																										
Stage Accelaration Max																										
Stage Diameter Max																										
Stage Length																										
Stage Thrust																										
Developer																										
Main Developer																										
Main Manufacturer																										
Manufacturer																										
Prime Contractor																										
Subcontractor																										

Tab. 5-9a Abhängigkeitsregeln der Oberklasse `Stage`

	Baufaktor	Growth Factor	Overall Mass Ratio	Structure Ratio	Actuator System Mass	Burn-Out Mass	Dry Mass	Groß Mass	Guidance and Control Mass	Lift-Off Mass	Net Mass	Propulsion System Mass	Recovery System Mass	Structure Mass	Stage Accelaration Max	Stage Diameter Max	Stage Length	Stage Thrust	Developer	Main Developer	Main Manufacturer	Manufacturer	Prime Contractor	Subcontractor
Pressurization Gas								x		x				x	x	x	x	x						
Propellant	x					x	x	x		x	x	x		x	x	x	x	x						
Tank								x		x				x	x	x	x	x						
Propellant Other						x	x	x		x	x	x		x	x	x	x	x						
Oxidizer Tank						x	x	x		x	x	x		x	x	x	x	x						
Fuel Tank						x	x	x		x	x	x		x	x	x	x	x						
Equipment Bay								x	x	x				x	x	x	x	x						
PL Interface								x		x				x	x	x	x	x						
Interstage Adapter								x		x				x	x	x	x	x						
Conceptual Design Start								x		x				x	x	x	x	x						
Conceptual Design End								x		x				x	x	x	x	x						
Prelimenary Design Start								x		x				x	x	x	x	x						
Prelimenary Design End								x		x				x	x	x	x	x						
Program/Project Definition Start								x		x				x	x	x	x	x						
Program/Project Definition End								x		x				x	x	x	x	x						
Development Start								x		x				x	x	x	x	x						
Development End								x		x				x	x	x	x	x						
Final Design Review								x		x				x	x	x	x	x						
First Ignition Test								x		x				x	x	x	x	x						
Final Burn Time Test								x		x				x	x	x	x	x						
Production Start								x		x				x	x	x	x	x						
First Flight Attempt								x		x				x	x	x	x	x						
Succesful Flight Demonstration								x		x				x	x	x	x	x						
Production End								x		x				x	x	x	x	x						
Last Flight								x		x				x	x	x	x	x						
Final Documentation/Report								x		x				x	x	x	x	x						
Baufaktor								x		x	x			x	x	x	x	x						
Growth Factor							x	x		x				x	x	x	x	x						
Overall Mass Ratio								x		x				x	x	x	x	x						
Structure Ratio							x	x		x				x	x	x	x	x						
Actuator System Mass								x		x				x	x	x	x	x						
Burn-Out Mass							x	x		x	x			x	x	x	x	x						
Dry Mass						x		x		x	x			x	x	x	x	x						
Groß Mass								x		x				x	x	x	x	x						
Guidance and Control Mass								x		x				x	x	x	x	x						
Lift-Off Mass								x		x				x	x	x	x	x						
Net Mass						x	x	x		x				x	x	x	x	x						
Propulsion System Mass								x		x				x	x	x	x	x						
Recovery System Mass								x		x				x	x	x	x	x						
Structure Mass								x		x	x			x	x	x	x	x						
Stage Accelaration Max								x		x				x	x	x	x	x						
Stage Diameter Max								x		x				x	x	x	x	x						
Stage Length								x		x				x	x	x	x	x						
Stage Thrust								x		x				x	x	x	x	x						
Developer								x		x				x	x	x	x	x		x				
Main Developer								x		x				x	x	x	x	x	x					
Main Manufacturer								x		x				x	x	x	x	x				x		
Manufacturer								x		x				x	x	x	x	x	x		x			
Prime Contractor								x		x				x	x	x	x	x					x	
Subcontractor								x		x				x	x	x	x	x						x

Tab. 5-9b Abhängigkeitsregeln der Oberklasse `Stage`

Attribute	Klassen																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Pressurization Gas																							x	x
Propellant						x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Tank						x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Propellant Other																						x		x
Oxidizer Tank																x	x	x	x	x	x	x	x	x
Fuel Tank													x	x	x	x	x	x	x	x	x	x	x	x
Equipment Bay											x	x									x		x	x
PL Interface																							x	x
Interstage Adapter										x					x					x	x			x
Conceptual Design Start																							x	x
Conceptual Design End																							x	x
Prelimenary Design Start																							x	x
Prelimenary Design End																							x	x
Program/Project Definition Start																							x	x
Program/Project Definition End																							x	x
Development Start																							x	x
Development End																							x	x
Final Design Review																							x	x
First Ignition Test																							x	x
Final Burn Time Test																							x	x
Production Start																							x	x
First Flight Attempt																							x	x
Succesful Flight Demonstration																							x	x
Production End																							x	x
Last Flight																								x
Final Documentation/Report																								x
Baufaktor						x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Growth Factor																								x
Overall Mass Ratio																								x
Structure Ratio																								x
Actuator System Mass																			x					x
Burn-Out Mass				x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Dry Mass	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Groß Mass	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Guidance and Control Mass											x	x									x		x	x
Lift-Off Mass	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Net Mass	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Propulsion System Mass						x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Recovery System Mass																								x
Structure Mass	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Stage Accelaration Max	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Stage Diameter Max	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Stage Length	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Stage Thrust	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Developer	x	x	x		x		x	x	x			x		x			x	x		x	x		x	
Main Developer		x	x					x	x			x		x				x			x		x	
Main Manufacturer	x		x		x		x		x			x		x			x	x		x	x		x	
Manufacturer	x		x		x		x		x			x		x			x	x		x	x		x	
Prime Contractor																							x	
Subcontractor																							x	

Tab. 5-10 Weitere Unterteilung der Oberklasse `Stage` (nach Anwendung der Regeln)

Engine (Triebwerk)

Da bei den Triebwerken die Anzahl der Objekte und die Anzahl der Eigenschaften am Größten ist, ergab sich erwartungsgemäß die größte Anzahl Klassen (über 700) vor der Nachbearbeitung. Dies warf insofern Probleme auf, als dass MS Excel maximal 254 Spalten bearbeiten kann und nicht in der Lage war die Anzahl Klassen darzustellen. Daher wurden die Gruppen bereits bei der Feststellung der vorhandenen Attributkombinationen in der Datenbank benutzt, um die Anzahl der Klassen zu reduzieren. Dadurch konnte die Anzahl der Kombinationen auf 208 gesenkt werden. Durch Anwendung der Regeln und Zusammenfassen der Gruppen wurde die Anzahl auf 64 Klassen gesenkt.

Die elektrischen Triebwerke werden im Expertensystem nicht behandelt, da am Institut die entsprechenden Experten fehlen und die elektrischen Triebwerke nur einen geringen Prozentsatz der gespeicherten Triebwerke ausmachen. Daher werden alle elektrischen Triebwerke zu einer einzigen Gruppe zusammengefasst. Alle Triebwerke, die mindestens eins der in Tab. 5-11 gezeigten Attribute besitzen, werden als elektrische Triebwerke eingestuft. Für diese Eigenschaften existieren auch keine Regeln. Allerdings werden die nicht aufgeführten Eigenschaften auch für elektrische Triebwerke durch Regeln erzeugt, wenn der Klassifikator feststellt, dass sie sie besitzen.

Die Gruppe „Engine“ in Tab. 5-12 enthält Attribute die immer vorhanden sein müssen. Auch bei den Triebwerken gilt: Die Gruppen, die Subsysteme kennzeichnen sind Kandidaten für Entity-Tabellen. Aber auch hier macht die Speicherung in einer eigenen Tabelle nur Sinn, wenn die Subsysteme in mehreren Triebwerken Verwendung finden.

Acceleration Electrode Transparancy	Heater Current
Acceleration Grid Current	Heater Voltage
Acceleration Grid Holes Diameter	Interelectrod Spacing
Acceleration Power Losses	Ion Cost
Beam Current	Ion Cost Maximum
Beam Current Maximum	Ion Cost Minimum
Cathode Mass Flow	Ionising Power
Current Density	Neutralizer Power
Current Density Max	Propellant Utilisation
Current Density Min	Propellant Utilisation Maximum
Depth of Dishing	Propellant Utilisation Minimum
Discharge Current	Screen Electrode Transparancy
Discharge Voltage	Screen Grid Holes Diameter
Electric Power Efficiency	Screen Grid Holes Width
Electrical Power Input	Specific Discharge Energy
Electrical Power Input Max	Total Equivalent A Flow
Energy Consumption In Pulse	U+
Generator Frequency	U-

Tab. 5-11 Attribute elektrischer Triebwerke

Engine	Efficiency	Nozzle	Dichte der Abgase am Düsenende
	Engine Diameter		spezielle Gaskonstante
	Engine Height		Adiabatenexponent
	Engine Length		Van den Kerkhove-Faktor
	Engine Width		Expansion Area
	Exhaust Velocity		Expansion Ratio
	Mass Flow Rate		Nozzle Area Ratio
	Specific Impulse		Nozzle Area Ratio End
	Specific Impulse Sea Level		Nozzle Exit Area
	Specific Impulse Vacuum		Nozzle Exit Diameter
	Thrust		Nozzle Exit Pressure
	Thrust Coefficient		Nozzle Expansion Ratio
	Thrust Coefficient Sea Level		Nozzle Length
	Thrust Coefficient Vacuum		Nozzle Mass
	Thrust Sea Level		Nozzle Throat Area
	Thrust Vacuum		Nozzle Throat Diameter
	Total Efficiency		Nozzle Throat Pressure
	Total Mass		Nozzle Throat Velocity
Chamber	Chamber Combustion Pressure		halber Öffnungswinkel
	Chamber Combustion Temperature	Diergol Engine	Chamber Mixture Ratio
	Chamber Diameter		Chamber Oxidizer Inlet Pressure
	Chamber Fuel Inlet Pressure		Chamber Oxidizer Inlet Pressure Maximum
	Chamber Fuel Inlet Pressure Maximum		Chamber Oxidizer Inlet Pressure Minimum
	Chamber Fuel Inlet Pressure Minimum		Chamber Oxidizer Inlet Temperature
	Chamber Fuel Inlet Temperature		Oxidizer Mass Flow Rate
	Chamber Inlet Pressure		Mixture Ratio
	Chamber Inlet Pressure Maximum		Mixture Ratio Maximum
	Chamber Inlet Pressure Minimum		Mixture Ratio Minimum
	Chamber Length	O/F varied Engine	Engine Regulation Range (Lower Limit)
	Chamber Mass		Engine Regulation Range (Mixture Ratio)
	Chamber Wall Temperature Maximum		Engine Regulation Range (Upper Limit)
	Characteristic Length		Full Power Level
	Characteristic Velocity		Throttleability
	Convergent Chamber Mass		Throttling Maximum
	Cylindrical Chamber Mass		Throttling Minimum
	Divergent Chamber Mass	Solid Engine	Abbrandgeschwindigkeit
	Proof Pressure		Abbrandoberfläche
Fuel	Droplet Diameter		Aktionszeit
	Fuel Mass Flow Rate		Äussere Klemmung
	Feed System Mass		Webb-Thickness
	Dry Mass		Zündverzugszeit
	Wet Mass	Burn Engine	Burn Time
	Burnout Mass		Burn Time Maximum
	Treibstoffdichte		Burn Time Per Mission
Pulse Engine	Pulse Duration	Gimbaled Engine	Deflection Maximum
	Pulse Impulse		Gimbal Angle Maximum
	Response Time,10 % Thrust		Gimbaling
	Response Time,90 % Thrust	Lifetime Engine	Lifetime
	Time Off		Cycle Life
	Time On		Service Lifetime
	Impulse Bit Minimum	Heated Engine	Start Heating-Up Time
PCU	PCU Height		Starting Heaters Power
	PCU Length	Cooled Engine	Coolant Flow Rate
	PCU Width		Coolant Mass Flow Rate
	PCCS Mass		

Tab. 5-12 Attributgruppen der Oberklasse `Engine`

	Engine	Chamber Engine	Nozzle Engine	Fuel Engine	Diergol Engine	Solid Engine	Burn Engine	O/F varied Engine	Cooled Engine	PCU	Gimbaled Engine	Lifetime Engine	Heated Engine	Pulse Engine	Conceptual Design Start	Conceptual Design End	Prelimenary Design Start	Prelimenary Design End	Program/Project Definition Start	Program/Project Definition End	Development Start	Ignition Test/First Hot Firing	First Full Thrust Test	First Full Thrust & Full Burn Time Test	Qualification Test End	First Flight Test	Production Start	Production End	Last Flight	Final Documentation/Report
Engine	x																													
Chamber Engine	x		x				x																							
Nozzle Engine	x	x		x																										
Fuel Engine	x						x																							
Diergol Engine	x	x					x																							
Solid Engine	x	x	x				x																							
Burn Engine	x																													
O/F varied Engine	x				x																									
Cooled Engine	x																													
PCU	x																													
Gimbaled Engine	x																													
Lifetime Engine	x																													
Heated Engine	x																													
Pulse Engine	x																													
Conceptual Design Start	x																													
Conceptual Design End	x														x															
Prelimenary Design Start	x														x	x														
Prelimenary Design End	x														x	x	x													
Program/Project Definition Start	x														x	x	x	x												
Program/Project Definition End	x														x	x	x	x	x											
Development Start	x														x	x	x	x	x	x										
Ignition Test/First Hot Firing	x														x	x	x	x	x	x	x									
First Full Thrust Test	x														x	x	x	x	x	x	x	x								
First Full Thrust & Full Burn Time Test	x														x	x	x	x	x	x	x	x	x							
Qualification Test End	x														x	x	x	x	x	x	x	x	x	x						
First Flight Test	x														x	x	x	x	x	x	x	x	x	x	x					
Production Start	x														x	x	x	x	x	x	x	x	x	x	x	x				
Production End	x														x	x	x	x	x	x	x	x	x	x	x	x	x			
Last Flight	x														x	x	x	x	x	x	x	x	x	x	x	x	x	x		
Final Documentation/Report	x														x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
Controller Mass	x																													
Tube Mass	x																													
Heat Exchanger Mass	x																													
Jacket Mass	x																													
Manifold Mass	x																													
Number Of Restarts	x																													
PM Value	x																													
Poisson Ratio	x																													
Specific Mass Flow Rate	x																													
Specific Power	x																													
Thrust Maximum	x																													
Thrust Minimum	x																													
Thrust Vacuum Max	x																													
Thrust Vacuum Min	x																													
Main Designer	x																													
Developer	x																													
Manufacturer	x																													

Tab. 5-13a Abhängigkeitsregeln der Oberklasse `Engine`

	Controller Mass	Tube Mass	Heat Exchanger Mass	Jacket Mass	Manifold Mass	Number Of Restarts	PM Value	Poisson Ratio	Specific Mass Flow Rate	Specific Power	Thrust Maximum	Thrust Minimum	Thrust Vacuum Max	Thrust Vacuum Min	Main Designer	Developer	Manufacturer
Engine																	
Chamber Engine																	
Nozzle Engine																	
Fuel Engine																	
Diergol Engine																	
Solid Engine																	
Burn Engine																	
O/F varied Engine	x																
Cooled Engine																	
PCU																	
Gimbaled Engine																	
Lifetime Engine																	
Heated Engine																	
Pulse Engine																	
Conceptual Design Start															x		
Conceptual Design End																	
Prelimenary Design Start																	
Prelimenary Design End																	
Program/Project Definition Start																	
Program/Project Definition End																	
Development Start																x	
Ignition Test/First Hot Firing																	
First Full Thrust Test																	
First Full Thrust & Full Burn Time Test																	
Qualification Test End																	
First Flight Test																	
Production Start																	x
Production End																	
Last Flight																	
Final Documentation/Report																	
Controller Mass																	
Tube Mass																	
Heat Exchanger Mass																	
Jacket Mass																	
Manifold Mass																	
Number Of Restarts																	
PM Value																	
Poisson Ratio																	
Specific Mass Flow Rate																	
Specific Power																	
Thrust Maximum												x					
Thrust Minimum											x						
Thrust Vacuum Max														x			
Thrust Vacuum Min													x				
Main Designer																	
Developer															x		
Manufacturer																x	

Tab. 5-13b Abhängigkeitsregeln der Oberklasse `Engine`

	Klassen																													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Engine	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Chamber Engine																		x	x	x	x	x	x	x	x	x	x	x	x	x	x
Nozzle Engine																		x	x	x	x	x	x	x	x	x	x	x	x	x	x
Fuel Engine											x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x
Diergol Engine																															
Solid Engine																		x													x
Burn Engine		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
O/F varied Engine																															
Cooled Engine																															
PCU																															
Gimbaled Engine																															
Lifetime Engine																	x													x	
Heated Engine																															
Pulse Engine																x												x	x	x	
Conceptual Design Start						x	x	x	x	x				x	x									x	x	x	x				
Conceptual Design End						x	x	x	x	x				x	x									x	x	x	x				
Prelimenary Design Start						x	x	x	x	x				x	x									x	x	x	x				
Prelimenary Design End						x	x	x	x	x				x	x									x	x	x	x				
Program/Project Definition Start						x	x	x	x	x				x	x									x	x	x	x				
Program/Project Definition End						x	x	x	x	x				x	x									x	x	x	x				
Development Start						x	x	x	x	x				x	x									x	x	x	x				
Ignition Test/First Hot Firing								x	x	x					x										x	x	x				
First Full Thrust Test								x	x	x					x										x	x	x				
First Full Thrust & Full Burn Time Test								x	x	x					x										x	x	x				
Qualification Test End								x	x	x					x										x	x	x				
First Flight Test									x	x																	x				
Production Start																															
Production End																															
Last Flight																															
Final Documentation/Report																															
Controller Mass																															
Tube Mass																															
Heat Exchanger Mass																															
Jacket Mass																															
Manifold Mass																															
Number Of Restarts													x																		
PM Value																															
Poisson Ratio																															
Specific Mass Flow Rate																															
Specific Power																															
Thrust Maximum					x							x										x	x						x		
Thrust Minimum					x							x										x	x						x		
Thrust Vacuum Max																															
Thrust Vacuum Min																															
Main Designer	x		x	x		x	x	x	x	x	x	x		x	x	x	x			x	x		x	x	x	x	x	x	x	x	
Developer	x		x	x		x	x	x	x	x	x	x		x	x	x	x			x	x		x	x	x	x	x	x	x	x	
Manufacturer	x			x			x	x		x	x	x		x	x	x	x				x		x	x		x	x	x	x	x	

Tab. 5-14a Weitere Unterteilung der Oberklasse `Engine` (nach Anwendung der Regeln)

	Klassen																															
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
Engine		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Chamber Engine		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Nozzle Engine		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Fuel Engine		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Diergol Engine		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Solid Engine																																x	x
Burn Engine		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
O/F varied Engine																														x	x	x	x
Cooled Engine																													x				x
PCU																																	x
Gimbaled Engine																											x	x			x		x
Lifetime Engine																	x	x	x	x	x	x	x	x	x	x				x	x	x	
Heated Engine																																	
Pulse Engine															x	x								x	x	x							x
Conceptual Design Start						x	x	x	x	x	x	x	x	x				x	x	x	x	x	x					x	x	x			x
Conceptual Design End						x	x	x	x	x	x	x	x	x				x	x	x	x	x	x					x	x	x			x
Prelimenary Design Start						x	x	x	x	x	x	x	x	x				x	x	x	x	x	x					x	x	x			x
Prelimenary Design End						x	x	x	x	x	x	x	x	x				x	x	x	x	x	x					x	x	x			x
Program/Project Definition Start						x	x	x	x	x	x	x	x	x				x	x	x	x	x	x					x	x	x			x
Program/Project Definition End						x	x	x	x	x	x	x	x	x				x	x	x	x	x	x					x	x	x			x
Development Start						x	x	x	x	x	x	x	x	x				x	x	x	x	x	x					x	x	x			x
Ignition Test/First Hot Firing							x	x	x	x	x	x	x	x						x	x	x	x					x	x	x			x
First Full Thrust Test							x	x	x	x	x	x	x	x						x	x	x	x					x	x	x			x
First Full Thrust & Full Burn Time Test							x	x	x	x	x	x	x	x						x	x	x	x					x	x	x			x
Qualification Test End							x	x	x	x	x	x	x	x						x	x	x	x					x	x	x			x
First Flight Test										x	x	x	x	x								x	x					x	x	x			x
Production Start																																	x
Production End																																	x
Last Flight																																	x
Final Documentation/Report																																	x
Controller Mass					x				x					x							x								x	x	x	x
Tube Mass																													x				x
Heat Exchanger Mass													x																				x
Jacket Mass																													x				
Manifold Mass																													x				x
Number Of Restarts								x				x							x		x		x							x		x	x
PM Value																										x							
Poisson Ratio																													x				
Specific Mass Flow Rate																																	x
Specific Power																																	x
Thrust Maximum				x												x									x	x						x
Thrust Minimum				x												x									x	x						x
Thrust Vacuum Max																																	x
Thrust Vacuum Min																																	x
Main Designer			x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Developer			x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Manufacturer			x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x

Tab. 5-14b Weitere Unterteilung der Oberklasse `Engine` (nach Anwendung der Regeln)

5.3.2 Beschreibung des Klassifikators

Nach Durchführung der in den vorhergehenden Abschnitten beschriebenen Arbeitsschritte hat man für die Oberklassen `Launcher`, `Stage` und `Engine` eine Unterteilung in eine bestimmte Anzahl Unterklassen vorliegen. Zu jeder Unterklasse gehört eine bestimmte Menge von Attributen. Diese Listen können aus MS Excel exportiert und in einen HyperCard Stack importiert werden. Dieses HyperCard Programm, genannt Klassifikator, kann für beliebige Objekte (z.Zt. nur Objekte der Objektklassen `Launcher`, `Stage` und `Engine`)

- die Menge, der in der Datenbank vorhandenen Attribute,
- die Menge der Attribute, die dieses Objekt besitzen muss und
- die Menge der Attribute, die diesem Objekt fehlen

feststellen. Damit fungiert der Klassifikator als Bindeglied zwischen normalem Datenbank-Benutzerinterface und Expertensystem. Es sind zwei grundsätzliche Arten von Anfragen an das Informations- und Expertensystem durch einen Benutzer möglich:

- Der Benutzer möchte einen bestimmten Attributwert eines Objekts
- Der Benutzer möchte alle Attributwerte eines Objekts

In Zukunft sind natürlich auch folgende Zwischenlösungen denkbar:

- Der Benutzer möchte eine Liste von Attributwerten eines Objekts
- Der Benutzer möchte einen Attributwert von einer Liste von Objekten
- Der Benutzer möchte eine Liste von Attributwerten von einer Liste von Objekten
- usw.

Bei allen diesen Anfragen muss zunächst festgestellt werden, ob ein Attribut, dessen Wert gesucht wird, überhaupt eine Eigenschaft des betreffenden Objekts ist. Wenn dies der Fall ist, und wenn sich der oder die Werte nicht in der Datenbank finden lassen, wird versucht die fehlenden Attributwerte durch das Expertensystem bestimmen zu lassen. Gerade die Anfrage nach allen Eigenschaften eines Objekts ist die Standardanfrage an das Informationssystem. Die Aufgabe des Klassifikators besteht nun darin eine Liste aller Eigenschaften zu liefern, die dieses Objekt (oder bei einer Liste von Objekten auch für jedes Objekt der Liste) besitzen muss.

In Abb. 5-11 ist die Vorgehensweise um dies zu erreichen schematisch dargestellt. Der Klassifikator erhält als Eingangsinformation den Objekttyp (`Launcher`, `Stage` oder `Engine`) und die eindeutige Identifikationsnummer (`Launcher_ID`, `Stage_ID` oder `Engine_ID`), die das Objekt in der Datenbank identifiziert. Dann wird über eine SQL-Abfrage eine Liste mit allen zu diesem Objekt in der Datenbank vorhandenen Attributen erstellt. Die Attributlisten aller Klassen dieses Objekttyps sind dem HyperCard Programm durch Import aus der MS Excel Datei bekannt. Durch Vergleich mit diesen Attributlisten wird eine Liste möglicher Klassen erzeugt. Wenn die Liste der in der Datenbank vorhandenen Attribute eine Teilmenge der Liste der Attribute einer Klasse ist, dann ist das zu untersuchende Objekt möglicherweise Mitglied dieser Klasse, und die Klasse kommt in die Liste möglicher Klassen. Im nächsten Schritt wird die Schnittmenge aus allen Attributlisten der möglichen Klassen gebildet, um die Liste aller Attribute zu erhalten, die diesen Klassen gemeinsam ist. Dies ist auch gleichzeitig die Liste aller Attribute, die das zu untersuchende Objekt besitzen muss. Diese Liste kann identisch mit der Liste der vorhandenen Attribute, aber niemals eine Teilmenge dieser Liste, sein. Im letzten Schritt wird die Differenzmenge zwischen der Liste der in der Datenbank vorhandenen Attribute und der Liste der Attribute, die das Objekt besitzen muss, gebildet. Als Ergebnis erhält man eine Liste der Attribute, die dieses Objekt besitzen muss, welche aber in der Datenbank nicht vorhanden sind. Alle drei Attributlisten (Liste der in der Datenbank vorhandenen Attribute, Liste der Attribute, die das Objekt besitzen muss und Differenz der beiden Listen) können als Ergebnis ausgegeben werden.

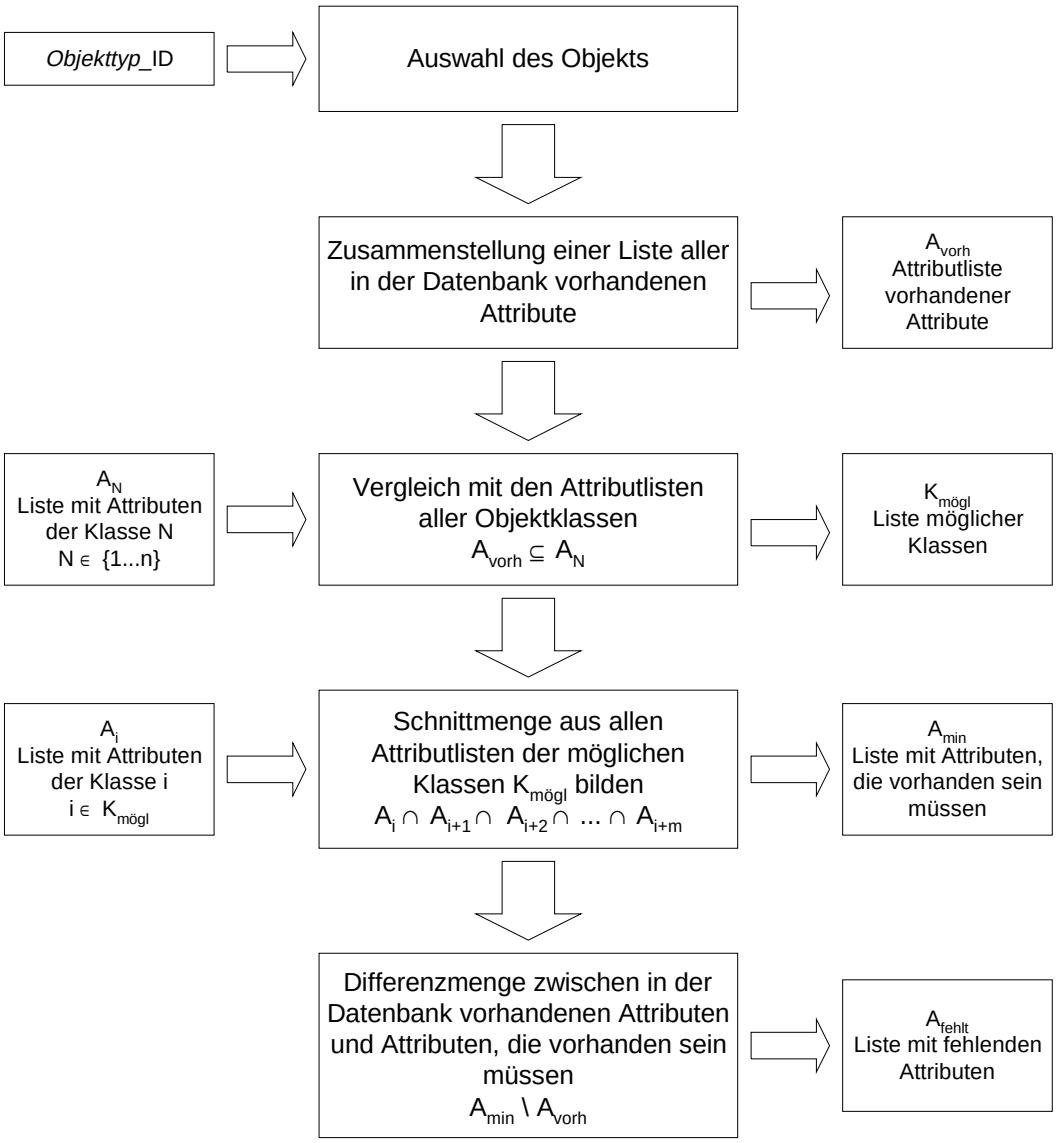


Abb. 5-11 Ablaufschema zum Auffinden von Lücken im Faktenwissen

Eine Alternative zu dem beschriebenen Vorgehen wäre die explizite Programmierung der Regeln für jeden Objekttyp in PROLOG. Dem PROLOG Klassifikator würde dann ebenfalls der Objekttyp und die Identifikationsnummer als Eingangsinformation übergeben werden. Der PROLOG Klassifikator besorgt sich die Liste aller vorhandenen Attribute aus der Datenbank und wendet die entsprechenden Regeln auf diese Liste an. Jede anwendbare Regel ergänzt die Liste um die Attribute, die aufgrund der Regel vorhanden sein müssen. Als Ergebnis erhält man eine Liste mit allen Attributen, die das jeweilige Objekt besitzen muss. Diese Vorgehensweise ist zwar im Hinblick auf die anzuwendenden Regeln transparenter für den Benutzer (man könnte sogar soweit gehen eine eigene Erklärungskomponente zu programmieren), ist jedoch bei einer großen Anzahl von Regeln entsprechend langsam in der Ausführung. Der Vergleich von Listen ist in HyperCard über sogenannte "External Commands" XCMD und "External Functions" XFNC welche in C oder Pascal programmiert sind, sehr bequem zu implementieren und vor allem schnell in der Ausführung. Hinzu kommt, dass die Klassenstruktur für den Benutzer nicht mehr sichtbar wäre, da nur die Regeln in PROLOG explizit sind; eine Einordnung in eine bestimmte Klasse mit den entsprechenden Eigenschaften findet nicht mehr statt. Außerdem ist die Speicherung der Regeln in Form von Excel Tabellen sehr übersichtlich und leicht zu ändern. Aus diesen Gründen wurde die Variante mit HyperCard als Klassifikator gewählt.

6 Expertensystem zur Ergänzung und Kontrolle des Faktenwissens

Während der Erstellung der relationalen Datenbank fiel auf, dass die Datensätze der meisten Raumfahrtobjekte entweder unvollständig waren oder falsche Daten enthielten. Da es für eine sinnvolle Anwendung der Daten aber notwendig ist, komplette und konsistente Datensätze zu erhalten, werden im AIM System die Daten durch Regeln und Expertenwissen ergänzt und überprüft. Hierzu wird eine Regelbasis in PROLOG benutzt, die über HyperCard Stapel als Benutzeroberfläche gesteuert wird.

Die Regelbasis besteht zunächst aus Regeln, die Wissen über die Struktur der Datenbank enthalten. Zusätzlich sind zwei weitere Kategorien von Regeln enthalten: Regeln welche Attributwerte liefern und Regeln, welche Bereiche für Attribute festlegen. Die Regelbasis selbst besteht aus einem PROLOG Programm, welches in verschiedene Dateien aufgeteilt alle Regeln enthält. Die Regeln mit dem Wissen über die Struktur der Datenbank können über ein HyperCard Programm automatisch erzeugt werden. Die Regeln, welche Attributwerte liefern, müssen wegen des Fehlens einer Wissensakquisitionskomponente per Hand in PROLOG programmiert werden. Als Ergänzung zu dem PROLOG Programm gibt es eine Kopie aller Regeln in der AIM Datenbank, um auf Textinformationen für die Erklärungskomponente zurückgreifen zu können. Diese Textinformationen werden von der Benutzeroberfläche verwendet um die Schlussfolgerungen des Expertensystems für den Benutzer transparent zu machen.

6.1 Übersicht

Die für ein einwandfreies Funktionieren notwendigen Komponenten des Expertensystems sind in Abb. 6-1 dargestellt. Alle Dateien und Programme müssen sich in einem Ordner befinden. In den nachfolgenden Absätzen werden die Komponenten im einzelnen beschrieben. Einige der Dateien werden später ausführlicher erläutert. Das gesamte Expertensystem läuft auf einem Apple Power PC unter LPA MacProlog und HyperCard. Die Anbindung an den Datenbankserver kann sowohl über TCP/IP als auch über Ethertalk bzw. AppleTalk erfolgen. Auf die dafür notwendigen Komponenten wird in Kap. 1 eingegangen.

HyperCard

Die grafische Benutzeroberfläche des Expertensystems und einige der Hilfsprogramme wurden aus Zeitgründen in HyperCard auf einem Apple Macintosh Computer erstellt. HyperCard eignet sich hervorragend für eine schnelle Programmentwicklung in der Testphase (rapid prototyping). Bei komplexeren Programmieraufgaben ist jedoch eine Compiler gestützte Programmiersprache vorzuziehen. Ein Vorteil von HyperCard ist jedoch die leichte Einbindung in eine WWW Umgebung. HyperCard eignet sich als CGI Skriptsprache für die Erstellung einer Zugriffsmöglichkeit über das WWW. Dieser Weg wurde bereits bei der Zugriffssoftware für den Datenbankteil des Informationssystems beschritten und hat sich bestens bewährt. Damit wird auch der Nachteil, das HyperCard nur auf Apple Computern zur Verfügung steht, wieder aufgewogen.

Programme

Der HyperCard Stack "Prolog Knowledge Creator" ist ein Werkzeug zur Erzeugung von Textdateien, die in PROLOG importiert werden können. Als Ergebnis erhält man PROLOG Fakten, die für die Objekttypen `Launcher, Stage` und `Engine` Informationen über die Datenbankstruktur enthalten.

Der "Klassifikator" wurde bereits ausführlich in Kap. 1 beschrieben.

Der HyperCard Stack "Prolog Einbindung" stellt die Benutzeroberfläche des Expertensystems dar. Mit einigen Modifikationen kann er auch als CGI Interface für eine Benutzung des Expertensystems über WWW dienen. Man kann ein Objekt und eine Eigenschaft auswählen und erhält den Attributwert als Antwort zurück. Zusätzlich dient das Programm auch als Erklärungskomponente, welche die Antworten des Expertensystems erläutert und den Lösungsweg strukturiert ausgeben kann.

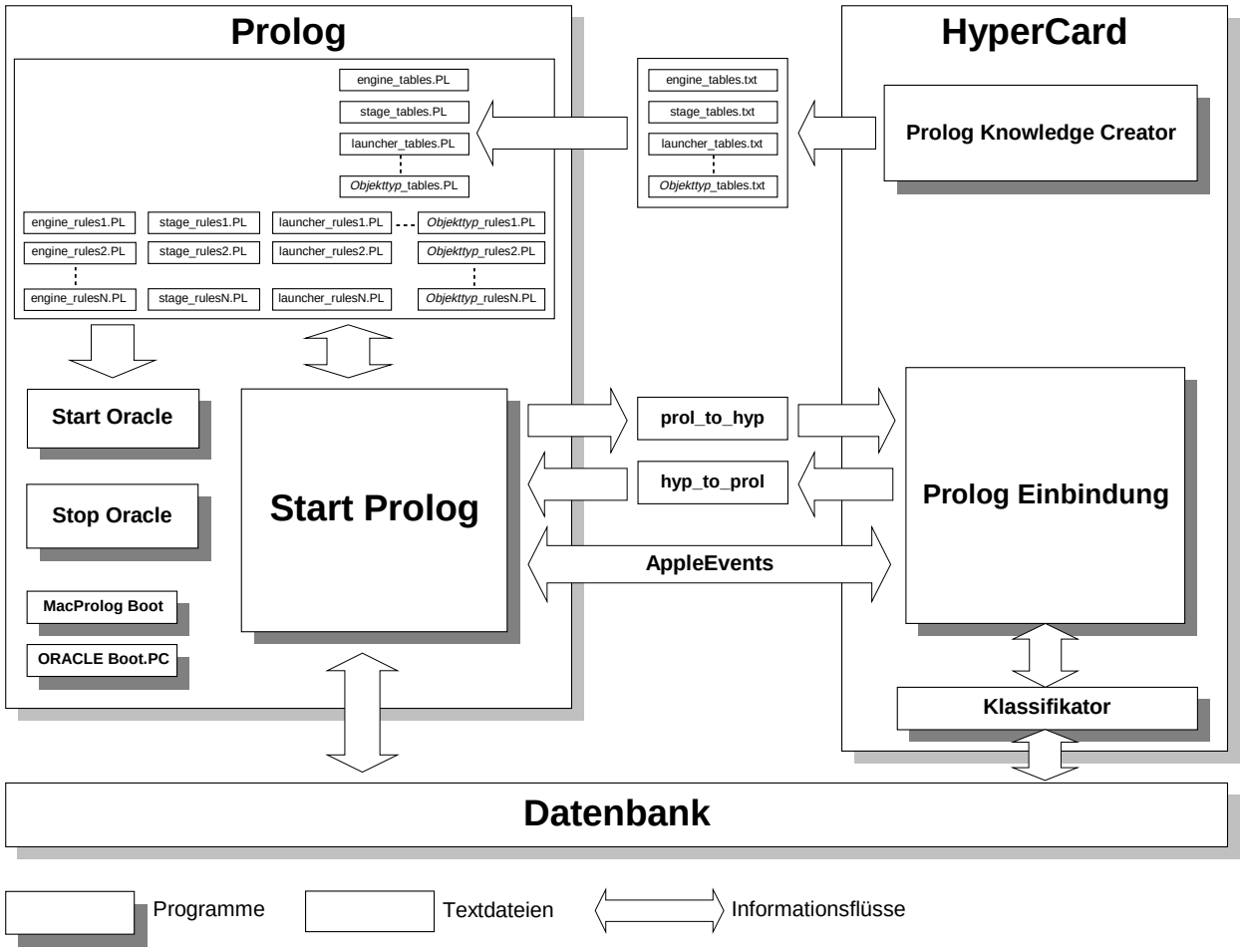


Abb. 6-1 Aufbau des Expertensystems

PROLOG

PROLOG bildet das eigentliche Expertensystem. MacProlog wurde als Programmiersprache gewählt, da eine Schnittstelle zur ORACLE Datenbank verfügbar war und eine Integration in die HyperCard Oberfläche einfach zu realisieren schien. Außerdem verfügt das Fachgebiet über einen erfahrenen PROLOG Programmierer, der bei der Entwicklung mit einbezogen wurde.

Programme

Das PROLOG Programm "Start Oracle" muss zum Start des Expertensystems einmal ausgeführt werden. Es stellt eine Verbindung mit der Datenbank her und lädt alle notwendigen PROLOG Programme in den Speicher. Dabei wird davon ausgegangen, dass der ORACLE Server bereits gestartet worden ist. Das komplette Listing ist im Anhang zu finden.

Das `'<LOAD>'(D)` Prädikat sorgt dafür, dass das Programm automatisch ausgeführt wird, wenn es in PROLOG geladen wird. Das Prädikat `ensure_loaded('Oracle Boot.PC')` stellt sicher, dass die Datei "Oracle Boot.PC" in PROLOG geladen wurde. Sie wird für den direkten Zugriff auf die Datenbank von PROLOG aus benötigt. Vor dem Einloggen in die Datenbank werden eventuell noch bestehende Sitzungen durch das Prädikat `db_close` geschlossen. Danach verbindet sich PROLOG mit der Datenbank durch Aufruf des Prädikats `db_open` mit den Parametern Benutzername (in diesem Fall "AIM") und Kennwort (das aktuelle Kennwort des Benutzers "AIM"). Im Anschluss wird das `'<LOAD>'` Prädikat durch Aufruf des `abolish` Prädikat gelöscht, da sonst eine Fehlermeldung bei nochmaliger Benutzung des `'<LOAD>'` Prädikats den Programmfluss stoppen würde .

Nach diesen Vorbereitungen werden die Regeln mit Hilfe des `consult` Prädikats in PROLOG geladen. Abschließend wird das "Start Oracle" Fenster mit dem Prädikat `wkill` geschlossen um Speicherplatz freizugeben.

Das PROLOG Programm "Stop Oracle" muss vor Beendigung des PROLOG Systems aufgerufen werden, da es sonst zum Rechnerabsturz kommt. Einzige Aufgabe des Programms ist die saubere Trennung der ORACLE Verbindung durch Aufruf des Prädikats `db_close`. Das komplette Listing findet sich ebenfalls im Anhang.

Kommunikation PROLOG ↔ HyperCard

Der "Prolog Knowledge Creator" erzeugt die Textdateien "*Objekttyp*_tables.txt" welche in PROLOG Programme umgewandelt werden durch ändern der FileCreator_ID auf "SIGM" mit dem Programm ResEdit und hinzufügen der Endung ".PL". Das Hinzufügen der Endung ist nicht unbedingt nötig, kennzeichnet die Dateien aber als PROLOG Dateien. Dadurch erhält man die PROLOG Dateien "*Objekttyp*_tables.PL". Der Aufbau dieser Dateien wird in Kap. 6.3 ausführlich beschrieben.

Für den Datenaustausch wurde zunächst eine nur auf AppleEvents basierende Lösung versucht. AppleEvents sind eine ereignisorientierte Möglichkeit der Interapplication Communication IAC auf Apple Computern. Mit ihrer Hilfe ist es möglich, das Programme auch über das Netzwerk miteinander kommunizieren. Da MacProlog jedoch nur sehr einfache AppleEvents versteht gestaltete sich der Austausch von Daten sehr schwierig. Die Übertragung von größeren Datenmengen mit Hilfe der IAC erwies sich als unmöglich. Da aber die Antworten des Prologteils des Expertensystems an die HyperCard Oberfläche sehr umfangreich ausfallen können, wurde für den Datenaustausch auf Textdateien zurückgegriffen. Die Dateien "prol_to_hyp" und "hyp_to_prol" dienen dem Datenaustausch zwischen PROLOG und HyperCard. "hyp_to_prol" enthält die Anfrage an das Expertensystem, welche vom PROLOG Programm beantwortet wird, und "prol_to_hyp" enthält die Antwort von PROLOG an die HyperCard Oberfläche.

Für die Ablaufsteuerung wurden jedoch einfache AppleEvents eingesetzt. Sowohl HyperCard als auch PROLOG starten sich gegenseitig über einen launch_Application AppleEvent. HyperCard generiert die Textdatei hyp_to_prol, die folgende PROLOG Klausel enthält:

```
'<LOAD>'(D):- hyp_asks_prolog(Objekttyp,Attribut,Objekttyp_ID[, Anzahl]).
```

Danach wird mit dem HyperCard Befehl:

```
open "hyp_to_prol" with "MacProlog32o"
```

diese Datei mit PROLOG geöffnet. Das `'<LOAD>'` Prädikat bewirkt eine automatische Ausführung des Programms.

Der Aufruf des Prädikats hyp_asks_prolog mit den Parametern

- Objekttyp: launcher, stage oder engine (Kleinschreibung beachten)
- Attribut: Eine Attributbezeichnung des entsprechenden Objekttyps (in gerade Anführungszeichen setzen, z.B. 'Net Mass'
- Objekttyp_ID: Der Primärschlüssel der entsprechenden Entity-Tabelle der AIM Datenbank
- Anzahl: Die Angabe der Anzahl an gewünschten Lösungen ist optional. Wird keine Angabe gemacht werden alle Lösungen gesucht. Wird eine Zahl angegeben, wird nur eine Lösung gesucht, egal welche Zahl angegeben wurde.

startet die Problemlösungskomponente des Expertensystems. Das PROLOG Programm "Start Prolog" sollte geladen sein, da sich die Definition des Prädikats in dieser Datei befindet. Ein komplettes Listing von "Start Prolog" befindet sich im Anhang. Die Problemlösungskomponente wird in Kap 6.3 ausführlich beschrieben.

PROLOG versucht nun mit Hilfe der in den Dateien *Objekttyp*_rules.PL definierten Regeln eine oder alle Lösungen für das angegebene Objekt zu finden. Das PROLOG Programm erzeugt die Datei "prol_to_hyp" die in Textform die Lösung(en) und Lösungsweg(e) enthält:

```
[10, kg, [Net Mass,0,8]]
[10, kg, [Net Mass,750,[Burn-Out Mass,0,8]]]
[0, kg, [Net Mass,469,[Lift-Off Mass,0,8],[Usable Propellant Mass,0,8]]]
```

Jede Zeile der Datei entspricht einer gefundenen Lösung. Wird keine Lösung gefunden, wenn z.B. keine Regel anwendbar ist und die Datenbank keine der gesuchten Daten enthält, wird nur die Textzeile "no solution" übergeben. Danach wird mit Aufruf des Prädikats

```
launch_app('HyperCard',switch,continue,PSN)
```

an HyperCard übergeben. Durch den Parameter `switch` wird HyperCard das aktive Fenster und durch `continue` wird die Ausführung des PROLOG Programms nicht unterbrochen.

HyperCard fängt AppleEvents mit folgendem EventHandler ab:

```
on appleEvent class, ID, sender
 set itemDelimiter to colon
 if last item of sender = "MacProlog32o" then
 send mouseUp to btn "Compute PROLOG Answer"
 end if
 pass appleEvent
end appleEvent
```

Dieser Handler überprüft lediglich, ob der gerade empfangene AppleEvent von PROLOG gesendet wurde. Ist dies der Fall wird der `mouseUp` Handler des HyperCard Buttons "Compute Prolog Answer" gestartet. Dieser wiederum wertet die PROLOG Lösungen aus und bereitet die übersichtliche Darstellung der gefundenen Lösung(en) vor.

6.2 Wissensbasis

Als Wissensbasis für das Expertensystem zur Ergänzung des Faktenwissens dient die in Kap. 1 beschriebene ORACLE AIM Datenbank. Alle in der AIM Datenbank gespeicherten Attributwerte werden als Faktenwissen für die Problemlösungskomponente benutzt. Da die Struktur der Datenbank bewusst so gewählt wurde, dass die Möglichkeit besteht zu jedem Attribut beliebig viele Werte einzutragen, Computerprogramme und Benutzer aber in der Regel nur an einem Wert pro Attribut interessiert sind, mussten Regeln festgelegt werden, die sicherstellen, dass nur ein Wert pro Attribut aus der Datenbank ausgewählt wird.

Für diesen Zweck stehen verschiedenen Möglichkeiten zur Auswahl. Für eine Bewertung der Verlässlichkeit der in der Datenbank eingetragenen Daten stehen zwei Informationen zur Verfügung. Die Attributwerte sind in den in Kap. 4.2 beschriebenen Datentabellen gespeichert. Diese enthalten in der Spalte *`Objekttyp_Attributgruppe`*`Default` (z.B. die Spalte `Stage_GeometryDefault` der Tabelle `Stage_Geometry`) einen Eintrag für die Verlässlichkeit des Werts. Außerdem kann durch den Eintrag in der Spalte `Reference_ID` der Datentabellen die Quelle aus der die Daten stammen spezifiziert werden. Anhand der Informationen über die Quellen aus der die Daten stammen können nun Regeln definiert werden, die z.B.

- Daten aus neuen Quellen bevorzugen,
- Daten aus bestimmten, vertrauenswürdigen Quellen bevorzugen,
- mehrere Daten möglichst nur aus einer oder aus möglichst wenigen Quellen nehmen.

Dies hätte jedoch eine Bewertung der Quellen und andere zusätzliche Arbeiten erfordert. Auch wäre die Abfrage und damit die Verarbeitungsgeschwindigkeit verlangsamt worden, so dass die Einträge in der `Default` Spalte herangezogen wurden. Der Eintrag in der `Default` Spalte sollte zunächst einen

beliebigen Wahrscheinlichkeitswert zwischen 0 und 1 enthalten, wurde aber dann auf die Werte 0 (Eintrag = n) und 1 (Eintrag = y) beschränkt. Diese Information wird nun bei der Abfrage des Faktenwissen aus der Datenbank berücksichtigt.

Zusätzlich zu diesen beiden Optionen besteht die Möglichkeit vorhandene Daten durch Regeln zu überprüfen, so dass nicht konsistente Zahlenwerte erkannt und entsprechend markiert werden können. Diese Möglichkeit wurde aus Zeitgründen fallengelassen.

Einbindung ORACLE – PROLOG

Wenn das Expertensystem eine Anfrage (in PROLOG auch Query genannt) erhält wird zunächst versucht diese über das in der AIM Datenbank gespeicherte Faktenwissen zu beantworten. Hierzu wird nach dem entsprechenden Attributwert in der Datenbank gesucht. Dank des PROLOG ORACLE Interface (MacDBI /20/) ist es u.a. möglich SQL-Kommandos direkt in PROLOG auszuführen und die Ergebnisse als PROLOG Variablen zu verarbeiten.

Wissen über die Struktur der Datenbank

Zusätzlich zu den "normalen" Expertenregeln gibt es Klauseln, die dem Expertensystem Wissen über die Struktur der Datenbank vermitteln. Dies sind die festgelegten Namen der Attribute, die Tabellen in denen sich bestimmte Attribute befinden, sowie der Aufbau (Spaltennamen) der einzelnen Tabellen. Diese Klauseln werden von einem HyperCard Programm automatisch aus den Data Dictionary Tabellen der ORACLE Datenbank generiert. Die Attributlisten werden ebenfalls durch das HyperCard Programm erzeugt. Die so entstandenen Textfiles mit den Regeln können direkt über die LOAD/1 Funktion von PROLOG in bestehende PROLOG Programme eingebunden werden.

Um eine SQL-Abfrage für einen Attributwert an die AIM Datenbank zu stellen müssen Informationen über die Struktur der Datenbank bekannt sein. Die eigentlichen Daten stehen, wie in Kap 4.2 beschrieben, in den Entity-Tabellen und in den Datentabellen. Bei den Entity-Tabellen bezeichnet der Spaltenname gleichzeitig den Attributnamen und die Attributwerte stehen jeweils in einer eigenen Spalte. Daher müssen für eine SQL-Abfrage

- der Name der Tabelle, in der der Attributwert steht,
- der Name der Spalte, in der der Attributwert steht

bekannt sein. Diese Informationen befinden sich als PROLOG Fakten in den Dateien `launcher_tables.PL`, `stage_tables.PL` und `engine_tables.PL`. Die Information ist als PROLOG Prädikat in der folgenden Form gespeichert:

`Objekttyp``_tabelle(Attribut, Tabelle, Spalte).`

Bei Aufruf des Prädikats in PROLOG wird die Attributbezeichnung übergeben und die Variablen Tabelle und Spalte werden mit den benötigten Informationen instanziiert. Ein Auszug aus der `engine_tables.PL` Datei ist im Folgenden beispielhaft gezeigt:

```
engine_tabelle('Name','Engine','Engine_Name').
engine_tabelle('Short Name','Engine','Engine_ShortName').
engine_tabelle('Suspension','Engine','ENGINE_SUSPENSION').
engine_tabelle('Status','Engine','ENGINE_STATUS').
engine_tabelle('Reuseability','Engine','ENGINE_REUSEABILITY').
engine_tabelle('Restartable','Engine','ENGINE_RESTART').
engine_tabelle('Chamber Cooling','Engine','ENGINE_CHAMBERCOOLING').
engine_tabelle('Propellant','Engine','PROPELLANT_ID').
```

Bei den Datentabellen werden mehr Informationen benötigt, da hier, bedingt durch die Tabellenstruktur, alle Attributwerte in einer Spalte stehen und nur durch die Attributbezeichnungen, welche ebenfalls in einer Spalte stehen, identifiziert werden. Da bei einigen Attributen die Einheit mit gespeichert ist, und es keine Einschränkungen hinsichtlich der Verwendung unterschiedlicher Einheiten für gleichartige

Attribute gibt, muss die Einheit zwecks Weiterverarbeitung mit abgefragt werden. Wie bereits erwähnt soll die Default-Spalte dafür verwendet werden immer nur einen Attributwert weiterzuverarbeiten, so dass auch diese Information verfügbar sein muss. Für die Datentabellen sind also die folgenden Information notwendig:

- der Name der Tabelle
- der Name der Spalte, in der der Attributwert steht
- der Name der Spalte, in der die Attributbezeichnung steht
- der Name der Spalte, in der die Einheit zu finden ist
- der Name der Spalte, in der der Default Eintrag vorgenommen wird

Das führt zu folgendem PROLOG Prädikat:

```
Objekttyp_tabelle(Attribut,Tabelle,Attributspalte,Attributwertspalte,Einheitenspalte,D
   efault-Spalte).
```

Bei Aufruf des Prädikats in PROLOG wird die Attributbezeichnung übergeben und die Variablen werden mit den benötigten Informationen instanziiert. Ein Auszug aus der `engine_tables.PL` Datei ist im Folgenden beispielhaft gezeigt:

```
engine_tabelle('Abbrandgeschwindigkeit','Engine_Performance','Engine_PerfType','Engine
   _PerfValue','Engine_PerfUnit','Engine_PerfDefault').
engine_tabelle('Abbrandoberfläche','Engine_Performance','Engine_PerfType','Engine_Perf
   Value','Engine_PerfUnit','Engine_PerfDefault').
engine_tabelle('Acceleration Grid
   Current','Engine_Performance','Engine_PerfType','Engine_PerfValue','Engine_PerfUnit
   ','Engine_PerfDefault').
engine_tabelle('Acceleration Power
   Losses','Engine_Performance','Engine_PerfType','Engine_PerfValue','Engine_PerfUnit'
   ,'Engine_PerfDefault').
engine_tabelle('Adiabatenexponent','Engine_Performance','Engine_PerfType','Engine_Perf
   Value','Engine_PerfUnit','Engine_PerfDefault').
engine_tabelle('Aktionszeit','Engine_Performance','Engine_PerfType','Engine_PerfValue'
   ,'Engine_PerfUnit','Engine_PerfDefault').
engine_tabelle('Beam
   Current','Engine_Performance','Engine_PerfType','Engine_PerfValue','Engine_PerfUnit
   ','Engine_PerfDefault').
```

SQL-Abfragen

Die Abfrage der Attributwerte erfolgt, wie immer bei relationalen Datenbanken, über SQL-Befehle. Die SQL-Kommandos können bei Verwendung des MacDBI direkt in PROLOG eingegeben werden. Für jeden Objekttyp sind drei grundsätzliche Arten von Abfragen vorgesehen. Zunächst die Abfrage der in den Entity-Tabellen gespeicherten Attributwerte, dann die Abfrage der in den Datentabellen gespeicherten Attributwerte und zusätzlich noch spezielle Abfragen, um implizit gespeicherte Attributwerte zu erhalten. Auf die letzte Kategorie wird in Kap 6.3 noch ausführlich eingegangen.

Die Abfrage der Attributwerte in den Entity-Tabellen ist relativ einfach, da pro Spalte nur ein Attributwert gespeichert ist. Das folgende Listing zeigt das entsprechenden PROLOG Prädikat am Beispiel des Objekttyps `stage`:

```
stage(Attribut,Stage_ID,Result,'-',[Attribut,1],_):-
 stage_tabelle(Attribut,Table,ColName),
 db_sql_select(['SELECT ',ColName,' FROM ',Table,' WHERE Stage_ID = ', Stage_ID],
   [Result]),
 not Result=[].
```

Die für die Abfrage benötigten Informationen (Attributbezeichnung: `Attribut` und Identifikationsnummer: `Stage_ID`) werden als Parameter übergeben. Das Ergebnis (`Result`), die Einheit (in diesem Fall keine Einheit `'-'`) und die für die Lösung verwendete Regel (in der Form: [geliefertes Attribut, Regel_ID]) werden innerhalb des Prädikats instanziiert und als Ausgabedaten an das aufrufende

Programm übergeben. Die für den Aufbau des SQL-Kommandos benötigten Informationen werden durch den Aufruf des Prädikats `stage_tabelle` beschafft. Das MacDBI Prädikat `db_sql_select` erlaubt die Ausführung des SQL-Befehls und das Ergebnis der SQL-Abfrage wird in der Variable `Result` zurückgeliefert. Die letzte Zeile stellt sicher, das bei einem Gelingen des Prädikats `db_sql_select` mit `Result` als leerer Menge (was passieren würde, wenn das SQL-Kommando fehlerfrei funktioniert, aber kein entsprechender Attributwert oder der Wert NULL in der Datenbank gespeichert ist) das Prädikat `stage` misslingt.

Bei den in den Datentabellen gespeicherten Werten sollen nun Regeln angewandt werden um sicherzustellen, dass immer nur ein Attributwert geliefert wird. In Abhängigkeit von der Default-Spalte soll entschieden werden, welcher Wert aus der Datenbank, für die weitere Verarbeitung genommen werden soll. Folgende Konstellationen sind denkbar:

- genau ein Attributwert ist mit "default = y" gekennzeichnet
- mehrere Attributwerte sind mit "default = y" gekennzeichnet
- es gibt keinen "default = y" Wert aber genau einen "default = n" Wert
- es gibt keinen "default = y" Wert aber mehrere "default = n" Wert

Diese Fälle werden mit drei PROLOG Regeln abgedeckt:

```
WENN Anzahl der Einträge ≠ 0 UND Anzahl der default=y Einträge = 0
   DANN Mittelwert aus allen Attributwerten bilden

WENN Anzahl der default=y Einträge = 1
   DANN Attributwert mit default=y nehmen

WENN Anzahl der default=y Einträge > 1
   DANN Mittelwert aus den Attributwerten mit default=y bilden
```

Wenn diese Regeln in der oben genannten Reihenfolge (wichtig!) in PROLOG als Prädikate programmiert werden, dann sind die genannten Fälle abgedeckt. Im folgenden sind die drei PROLOG Prädikate aufgelistet. Das Prädikat `stage_tabelle` holt die Informationen über die Tabellenstruktur der AIM Datenbank, `db_sql_select` übergibt die SQL Abfragen an die Datenbank. Das Prädikat `proceed` bildet den Mittelwert und das Prädikat `check_unit` übernimmt die Umrechnung in SI-Einheiten.

```
stage(Attribut,Stage_ID,Result,Unit,[Attribut,Regel,1007],_):-
   stage_tabelle(Attribut,Table,Type,Value,UnitCol,Default),
   db_sql_select(['SELECT COUNT(*) FROM ',Table,
       ' WHERE Stage_ID = ',Stage_ID,
       ' AND ',Type,' = ''',Attribut,''''], Y),
   not Y=[0],
   db_sql_select(['SELECT COUNT(*) FROM ',Table,
       ' WHERE Stage_ID = ',Stage_ID,'
       ' AND ',Type,' = ''',Attribut,'''
       ' AND ',Default,' = ''y'' '], X),
   X=[0],
   findall([OraValue,OraUnit],
   db_sql_select(['SELECT ',Value,',',UnitCol,' FROM ',Table,
       ' WHERE Stage_ID = ', Stage_ID ,
       ' AND ',Type,' = ''',Attribut,''''], [OraValue,OraUnit]),Liste),
   proceed(Attribut,Liste,Result,Unit,Regel).
```

```
stage(Attribut,Stage_ID,Result,Unit,[Attribut,Regel,1008],_):-
   stage_tabelle(Attribut,Table,Type,Value,UnitCol,Default),
   db_sql_select(['SELECT COUNT(*) FROM ',Table,
       ' WHERE Stage_ID= ', Stage_ID ,
       ' AND ',Type,' = ''',Attribut,''
       ' AND ',Default,' = ''y'' '], X),
   X=[1],
   db_sql_select(['SELECT ',Value,',',UnitCol,' FROM ',Table,
       ' WHERE Stage_ID = ', Stage_ID ,
       ' AND ',Type,' = ''',Attribut,''
       ' AND ',Default,' = ''y'' '], [OraValue,OraUnit]),
   check_unit(Attribut,OraValue,OraUnit,Result,Unit,Regel).

stage(Attribut,Stage_ID,Result,Unit,[Attribut,Regel,1009],_):-
   stage_tabelle(Attribut,Table,Type,Value,UnitCol,Default),
   db_sql_select(['SELECT ''yes'' FROM ',Table,
       ' WHERE Stage_ID = ', Stage_ID ,
       ' AND ',Type,' = ''',Attribut,'''
       ' AND ',Default,' = ''y''
       GROUP BY Stage_ID HAVING COUNT(*) > 1 '], X),
   X=[yes],
   findall([OraValue,OraUnit],
       db_sql_select(['SELECT ',Value,',',UnitCol,' FROM ',Table,
       ' WHERE Stage_ID = ', Stage_ID ,
       ' AND ',Type,' = ''',Attribut,'''
       AND ',Default,' = ''y'' '],
       [OraValue,OraUnit]),
       Liste),
   proceed(Attribut,Liste,Result,Unit,Regel).
```

Mittelwertbildung

Das Prädikat `proceed` ist für die Mittelwertbildung unter Beachtung der Einheiten der Attributwerte verantwortlich. Die eigentliche Umrechnung der Einheiten übernimmt das Prädikat `check_unit` das hier über den Umweg des Hilfsprädikats `proc` vor jeder Summierung aufgerufen wird. Das eigentliche Addieren der Zahlenwerte erledigt dann das Prädikat `sum`.

```
proceed(Attribut,Liste,W,Unit,Regel):-
    proc(Attribut,Liste,[[V,U]|Rneu],Unit,Regel),
    length([[V,U]|Rneu],Len),
    sum(0,[[V,U]|Rneu],Sum),
    W is Sum/Len.
proc(Attribut,[[V1,U1]|R],[[V,U]|Rneu],U,[Reg|Regel]):-
    check_unit(Attribut,V1,U1,V,U,Reg),
    proc(Attribut,R,Rneu,U,Regel).
proc(Attribut,[],[],U,[]).
sum(S0,[[V|U]|R],S):- % Berechnung derSumme einer Liste
    S0_neu is S0+V,
    sum(S0_neu,R,S).
sum(S,[],S).  % Abbruchbedingung für Summenberechnung
```

Einheitenumrechnung

Die Umrechnung der Einheiten übernimmt das Prädikat `check_unit`. Der allgemeine Aufbau ist:

```
check_unit(Attributbezeichnung, alter Wert, alte Einheit, neuer Wert , neue Einheit,
   Regel_ID).
```

Eingabeparameter sind der Attributwert, die vorhandenen Einheit und die gewünschte Einheit. In einigen Fällen wird auch die Attributbezeichnung übergeben. Als Ergebnis erhält man den umgerechneten Wert und die Regel mit der umgerechnet wurde. Ist keine Umrechnung notwendig wird die Regel 0 angewandt, welche den ursprünglichen Wert zurück liefert.

In der Regel reicht für die Umrechnung die Eingabe der beiden Einheiten zwischen denen umgerechnet werden soll aus. In der Raumfahrt gibt es jedoch den gewichtsspezifischen Impuls, der in Sekunden angegeben wird und in [m/s] umgerechnet werden kann. Daher muss man eine zusätzliche Regel definieren, die von [s] in [m/s] umrechnet, wenn die Attributbezeichnung einen spezifischen Impuls beschreibt.

```
% Druck
check_unit(_,Value,'N/m^2',Value,'N/m^2',0).
check_unit(_,Value,'bar',NewValue,'N/m^2',1901):- NewValue is Value * 100000.
check_unit(_,Value,'Pa',Value,'N/m^2',1902).
% Kraft
check_unit(_,Value,'N',Value,'N',0).
check_unit(_,Value,'kN',NewValue,'N',1903):- NewValue is Value * 1000.
check_unit(_,Value,'mN',NewValue,'N',1904):- NewValue is Value / 1000.
% Masse
check_unit(_,Value,'kg',Value,'kg',0).
check_unit(_,Value,'g',NewValue,'kg',1905):- NewValue is Value / 1000.
check_unit(_,Value,'t',NewValue,'kg',1906):- NewValue is Value * 1000.
% Zeit (Ausnahmen für spezifischen Impuls in s)
check_unit('Specific Impulse',Value,'s',NewValue,'m/s',1907):-
    NewValue is Value * 9.81,!.
check_unit('Specific Impulse Vacuum',Value,'s',NewValue,'m/s',1907):-
    NewValue is Value * 9.81,!.
check_unit('Specific Impulse Sea Level',Value,'s',NewValue,'m/s',1907):-
    NewValue is Value * 9.81,!.
% normale Zeitumrechnungen
check_unit(_,Value,'s',Value,'s',0).
check_unit(_,Value,'min',NewValue,'s',1908):- NewValue is Value / 60.
check_unit(_,Value,'h',NewValue,'s',1909):- NewValue is Value / 3600.
```

6.3 Problemlösungskomponente

Als Problemlösungskomponente wird PROLOG verwendet. Der Vorteil von PROLOG gegenüber anderen herkömmlichen Programmiersprachen ist der, das PROLOG bereits einen Problemlösungsalgorithmus enthält. Ein PROLOG Programm ist eine Sammlung von Fakten und Regeln. Die Fakten und Regeln werden auch Prädikate genannt. Ein Prädikat besteht entweder aus "Head" (Kopf) und "Tail" (Rumpf) oder nur aus dem Kopf. Wenn Kopf und Rumpf vorhanden sind werden diese durch ":-" getrennt (`head :- tail`). Das Programm wird gestartet, indem eine Anfrage (Query) oder Hypothese an PROLOG gestellt wird, die mit "yes" oder "no", also wahr oder falsch, beantwortet wird. PROLOG benutzt als Inferenzmethode Rückwärtsverkettung mit backtracking. Treten in einer Hypothese Variable auf, versucht PROLOG durch Instanziierung der Variablen mit Werten die Richtigkeit der Hypothese zu erreichen (matching).

Durch Abarbeitung der Unterziele (Aufruf der im "Tail" enthaltenen Prädikate), die ebenfalls wahr oder falsch sein können (man spricht auch von succeed/gelingen oder fail/fehlschlagen) in immer tiefere Verzweigungsschichten, wird versucht die Hypothese auf wahre Fakten zurückzuführen. Gelingt ein Lösungsweg nicht, wird ein sogenanntes backtracking durchgeführt. Das heißt der endgültig fehlgeschlagene Pfad wird zurückgegangen und eine andere Alternative ausprobiert. Dadurch kann ein Lösungsbaum komplett abgearbeitet werden. Dies muss nicht immer von Vorteil sein.

Das Aufstellen der Hypothese (Eingabe des Query) übernimmt das PROLOG Programm `start_prolog.PL`, in dem das PROLOG Prädikat `hyp_asks_prolog` definiert ist, welches wie bereits beschrieben von der Datei `hyp_to_prol` aufgerufen wird. Das Prädikat `hyp_asks_prolog` erhält die notwendigen Eingabedaten vom HyperCard GUI.

Es existieren vier Definitionen des Prädikats `hyp_asks_prolog`. Eine mit drei Parametern und eine mit vier Parametern. Wie bereits beschrieben dient der vierte Parameter der Entscheidung, ob eine Lösung oder alle möglichen Lösungen gesucht werden sollen. Zur Unterscheidung von Prädikaten gleichen Namens, aber unterschiedlicher Parameterzahl folgt die Anzahl der Parameter (auch "arity" genannt) dem Prädikatnamen durch "/" getrennt (man schreibt zum Beispiel: `hyp_asks_prolog/3`).

Wenn nach allen Lösungen für eine Hypothese gesucht werden soll wird `hyp_asks_prolog/3` aufgerufen, dessen Definition im Folgenden beschrieben wird:

```
hyp_asks_prolog(ObjectClass,Property,Identifier):-
    wkill(hyp_to_prol),
    retractall(used(A1,A2)), retractall(wanted(C1)),
    asserta(wanted(Property)),
    db_reset_cursors, db_flag(show_db_error,Old,off),
    forall(ObjectClass(Property,Identifier,Answer,Unit,Regel,
   -1),ausgabe(Answer,Unit,Regel) ),
    launch_app('HyperCard',switch,continue,PSN),!.
```

Die Eingabedaten sind der Objekttyp (`ObjectClass`), die Attributbezeichnung (`Property`) und die Identifikationsnummer des Objekts (`Identifier`), dessen Attributwert gesucht wird. Das Prädikat `wkill/1` schließt das Fenster. `retractall/1` und `asserta` werden im Zusammenhang mit der Vermeidung von ungewollten Rekursionen ausführlich erläutert. `db_reset_cursors` und `db_flag` bereiten die Datenbankzugriffe vor. Eine ausführliche Erklärung dieser beiden Prädikate findet sich im Abschnitt "Sonstige Anmerkungen". Die entscheidende Zeile ist das Prädikat `forall/2`, welches ermöglicht, das alle Lösungen für das in der Variablen `ObjectClass` enthaltene Prädikat gesucht werden. `ObjectClass` kann zur Zeit `launcher`, `stage` oder `engine` enthalten. Die gefundenen Lösungen werden mit dem Prädikat `ausgabe` in die Datei `prol_to_hyp` geschrieben. Zusätzlich wird in PROLOG selbst eine Ausgabe auf dem Bildschirm erzeugt, um eine gewisse Kontrolle über die gefundenen Lösungen zu haben und eventuelle Fehler zu beseitigen.

```
ausgabe(A,U,R):-
    write('['), write(A), write(', '), write(U), write(', '), write(R), writenl(']'),
    fcreate('prol_to_hyp',0),    % Output in Datei
    fopen('prol_to_hyp',1),
    output('prol_to_hyp'),
    write('['), write(A), write(', '), write(U), write(', '), write(R), writenl(']'),
    fclose('prol_to_hyp').
```

Sollte die erste Definition des Prädikats `hyp_asks_prolog/3` keine Lösung finden, also fehlschlagen, so wird die zweite Definition ausgeführt, die ebenfalls eine Datei `prol_to_hyp` erzeugt, in der sich aber nur die Mitteilung "no solution" befindet. Auch hier wird eine zusätzliche Meldung im PROLOG Fenster auf dem Bildschirm ausgegeben.

```
hyp_asks_prolog(ObjectClass,Property,Identifier):-
    writenl('no solution'),       % Normaler Output
    fcreate('prol_to_hyp',0),    % Output in Datei
    fopen('prol_to_hyp',1),
    output('prol_to_hyp'),
    writenl('no solution'),
    fclose('prol_to_hyp')
    launch_app('HyperCard',switch,continue,PSN).
```

Wenn nur eine Lösung gewünscht wird, wird ein zusätzlicher Parameter übergeben und das Prädikat `hyp_asks_prolog/4` ausgeführt. Der Ablauf ist nahezu mit dem von `hyp_asks_prolog/3` identisch. Allerdings fehlt hier das Prädikat `forall` und das Prädikat `ausgabe` wird nicht benötigt, da die entsprechenden Befehle direkt im Prädikat `hyp_asks_prolog/4` integriert sind. Auch hier gibt es eine zweite Definition des Prädikats für die Ausgabe der Meldung "no solution".

```
hyp_asks_prolog(ObjectClass,Property,Identifier,Anzahl):-
    wkill(hyp_to_prol),
    retractall(used(A1,A2)), retractall(wanted(C1)),
    asserta(wanted(Property)),
    db_reset_cursors, db_flag(show_db_error,Old,off),
    ObjectClass(Property,Identifier,Answer,Unit,Regel,-1),
    not Answer=[],
   % Normaler Output
    write('['), write(Answer), write(', '), write(Unit), write(', '), write(Regel),
   writenl(']'),
    fcreate('prol_to_hyp',0),    % Output in Datei
```

```
 fopen('prol_to_hyp',1),
    output('prol_to_hyp'),
    write('['), write(Answer), write(', '), write(Unit), write(', '), write(Regel),
   writenl(']'),
    fclose('prol_to_hyp'),!
    launch_app('HyperCard',switch,continue,PSN),!.
```

Keine Lösung gefunden:

```
hyp_asks_prolog(ObjectClass,Property,Identifier,_):-
    writenl('no solution'),       % Normaler Output
    fcreate('prol_to_hyp',0),    % Output in Datei
    fopen('prol_to_hyp',1),
    output('prol_to_hyp'),
    writenl('no solution'),
    fclose('prol_to_hyp')
    launch_app('HyperCard',switch,continue,PSN).
```

6.4 Regelbasis

Die Regelbasis besteht aus PROLOG Regeln, die für jeden Objekttyp in einer oder mehreren Dateien gespeichert sind, wie in Abb. 6-1 gezeigt. Da MacProlog in einer Datei maximal 32 kB speichern kann, ist es bei einer größeren Anzahl Regeln notwendig mehrere Dateien zu benutzen. Die Regeln können mathematische Gleichungen, Expertenschätzungen, Relationen, etc. sein. Jede Regel liefert einen Zahlenwert für genau ein Attribut. Dafür benötigt sie 0 bis n andere Attributwerte. Jeder Regel ist ein "confidence factor" von 0 bis 10 zugeordnet, der angibt wie vertrauenswürdig das Ergebnis der Regel ist. Dieser Faktor bestimmt die Reihenfolge der Abarbeitung unter PROLOG. Regeln mit hohem "confidence factor" werden zuerst abgearbeitet. Dadurch wird erreicht, wenn z.B. nur eine Lösung gesucht wird, das diejenige Lösung mit dem höchsten "confidence factor" geliefert wird.

Die Regeln müssen zur Zeit noch direkt in PROLOG eingegeben werden. Es wurde auf eine Wissensakquisitionskomponente verzichtet, da in der vorliegenden Version eine automatische Wissensakquisition nicht vorgesehen war und ein PROLOG Programmierer dem Fachgebiet für die Eingabe der Regeln dauerhaft zur Verfügung steht. Die derzeit vorhanden Regeln wurden im Rahmen einer Diplomarbeit /55/ erstellt und der gesamte Vorgang ist dort sehr ausführlich erläutert worden. Daher wird in dieser Arbeit auf eine detaillierte Beschreibung des "knowledge engineering" verzichtet.

Beschreibung der Regeln

Der Aufbau der Regeln ist prinzipiell immer gleich. Alle Regeln liefern genau einen Attributwert. Hierfür benötigen sie keinen, einen oder mehrere andere Attributwerte. Diese wiederum können als Fakten in der AIM Datenbank stehen oder über andere Regeln beschafft werden. Jede Regel besteht aus genau einer PROLOG Klausel. Diese Klausel wiederum besteht aus mehreren Unterzielen.

Der typische Aufbau der Expertenregeln in PROLOG Syntax ist in Abb. 6-2 gezeigt. Für jeden Objekttyp wurde aus Gründen der Übersichtlichkeit ein eigenes PROLOG Prädikat mit dem Namen des Objekttyps definiert. Zum gegenwärtigen Zeitpunkt sind dies die Prädikate `launcher`, `stage` und `engine`. Die Anzahl der Regeln für die einzelnen Objekttypen ist aus Tab. 6-1 ersichtlich. Die hohe Anzahl an Regeln für den Objekttyp `engine` erklärt sich unter anderem mit der relativ hohen Anzahl von Attributen die zu diesem Objekttyp gehören.

Objekttyp	Anzahl Regeln
Engine	245
Launcher	125
Stage	116

Tab. 6-1 Anzahl Regeln pro Objekttyp

Als Eingabeparameter werden das gesuchte Attribut, die Objektidentifikationsnummer, die gewünschte Einheit übergeben und eine Variable, welche die Identifikationsnummer der aufrufenden Regel enthält übergeben. Als Ergebnis wird der Zahlenwert des gesuchten Attributs und der Lösungsweg als PROLOG Liste zurückgeliefert.

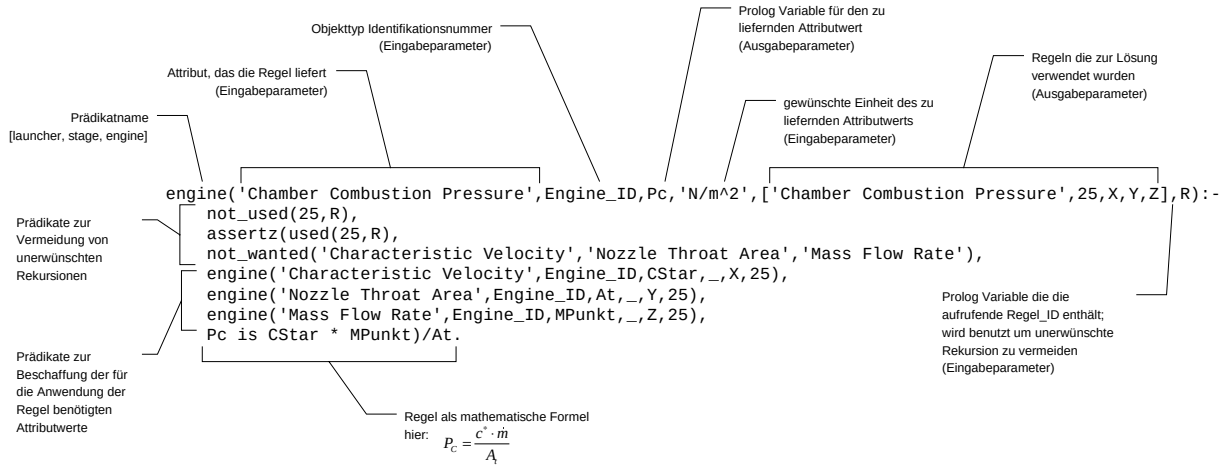


Abb. 6-2 Aufbau der Expertenregeln

Die ersten drei Subziele dienen der Vermeidung von unerwünschten Rekursionen und werden später noch detailliert beschrieben. Als nächstes werden die von der Regel benötigten Attributwerte beschafft. Hierfür werden die entsprechenden PROLOG Klauseln aufgerufen. Da die Regeln für den Zugriff auf die Datenbank sich nicht von den anderen Regeln unterscheiden und der Datenbankzugriff immer vor der Anwendung normaler Regeln erfolgt, wird zunächst versucht den benötigten Attributwert über eine Datenbankabfrage zu beschaffen. Gelingt dies nicht, sucht PROLOG nach der ersten Regel, die den benötigten Attributwert liefert und versucht diese anzuwenden. Diese Regel kann nun wiederum andere Attributwerte benötigen, welche wiederum zunächst in der Datenbank und dann über Regeln gesucht werden. Dadurch kann es zu ausgedehnten Rekursionen kommen an deren Ende aber immer ein Zugriff auf die Datenbank und somit auf das Faktenwissen steht. Umgekehrt betrachtet heißt das, dass basierend auf dem Faktenwissen der Datenbank Attributwerte durch Regeln erzeugt werden, mit denen wiederum andere Regeln angewendet werden können. Diese liefern solange neue Attributwerte, bis keine Regeln mehr zur Verfügung stehen oder keine Regeln mehr angewandt werden können, weil die Voraussetzungen für die Regeln nicht erfüllbar sind.

Als letztes Subziel innerhalb der Klausel steht die eigentliche Regel. Dies kann eine Formel sein (bei den derzeitigen Regeln ist dies hauptsächlich der Fall) oder eine andere Abschätzung, welche den gesuchten Attributwert berechnet oder liefert. In einigen Fällen stehen vor der eigentlichen Regel noch zusätzliche Sicherheitsabfragen, die dafür sorgen, dass bei der anschließenden Berechnung keine Laufzeitfehler auftreten. Dabei handelt es sich hauptsächlich um Abfragen, ob Terme, die in der nachfolgenden Formel im Nenner stehen, Null ergeben.

Die folgenden Tabellen (Tab. 6-2, Tab. 6-3, Tab. 6-4) zeigen Übersichten über die Anzahl von Regeln, die für jeden Objekttyp und jedes Attribut in der Regelbasis existieren. Ziel aller Versuche sollte es sein für jedes Attribut mindestens eine Regel zu finden. Es ist jedoch nicht immer möglich für alle Attribute eines Objekttyps Regeln zu definieren. In solchen Fällen sollte versucht werden mindestens Randwerte oder Maximal- bzw. Minimalwerte für eine grobe Abschätzung des Attributwertes zu finden.

Attribut	Anzahl
Lift-Off Mass	10
Dry Mass	6
Reuseability	6
Net Mass	6
Propulsion System Mass	5
Structure Mass	5
Recovery System Mass	5
Propellant Residuals Mass	4
Status	4
Coarse Structure Mass	3
Usable Extra Propellant Mass	3
Overall Mass Ratio	3
Guidance and Control Mass	3
Baufaktor, Conceptual Design End, Final Design Review, Last Flight, Prelimenary Design Start, Production Start, Production End, Usable Propellant Mass, Succesful Flight Demonstration, Program/Project Definition Start, Program/Project Definition End, Prelimenary Design End, Operation Cost, Groß Factor, First Flight Attempt, Development Start, Development End	2
Abolishment Cost, Launch Service Agency, Development Manager, DeltavLEO, DeltavGEO, Customer, Vehicle Amortization Cost, Structure Ratio, Number Of Boosters, Maximum Diameter, Manufacturing Cost, Maintenance And Refurbishment Cost, Life Cycle Cost, Launch Site And Range Cost, Launch Cost, Indirect Operation Cost, Propellant Ratio, Payload Ratio, Overall Length, Number of Launchers, Number Of Stages, Final Documentation/Report, Direct Operation Cost, Development Cost, Growth Factor, Conceptual Design Start, Average Launch Rate, Burn-Out Mass,	1
Customer, Developer, Development Manager, Launch Service Agency, Main Designer, Main Developer, Manufacturer, Vehicle Operator	0

Tab. 6-2 Anzahl Regeln pro Attribut für den Objekttyp `Launcher`

Attribut	Anzahl
Lift-Off Mass	10
Structure Mass	7
Propellant Residuals Mass	6
Propulsion System Mass	6
Dry Mass	5
Net Mass	5
Recovery System Mass	4
Pressurization Gas Mass	3
Conceptual Design End, First Flight Attempt, Fuel Tank Length, Fuel Tank Diameter, Program/Project Definition Start, Program/Project Definition End, Production Start, Production End, Prelimenary Design Start, Prelimenary Design End, Oxidizer Tank Volume, Oxidizer Tank Length, Oxidizer Tank Diameter, Last Flight, Ignition Test/First Hot Firing, Guidance and Control Mass, Usable Propellant Mass, Usable Extra Propellant Mass, Succesful Flight Demonstration, Stage Thrust, Fuel Tank Volume, Final Design Review, Development End, Development Start, Final Burn Time Test	2
Actuator System Mass, Baufaktor, Burn-Out Mass, Stage Diameter Max, Stage Accelaration Max, Propellant Total Mass, Propellant Ratio, Propellant Oxidizer Mass, Propellant Other Mass, Propellant Fuel Mass, Overall Mass Ratio, Tank Diameter, Structure Ratio, Interstage Adapter Mass, Growth Factor, Groß Mass, Final Documentation/Report, Equipment Bay Mass, Equipment Bay Diameter, Conceptual Design Start	1
Developer, Equipment Bay Length, Fuel Tank Pressure, Main Developer, Main Manufacturer, Manufacturer, Oxidizer Tank Pressure, PL Interface Diameter, Pressurization Gas Tank Pressure, Prime Contractor, Stage Length, Subcontractor, VEB Interface Diameter	0

Tab. 6-3 Anzahl Regeln pro Attribut für den Objekttyp `Stage`

Attribut	Anzahl
Chamber Combustion Pressure	52
Mixture Ratio	33
Mass Flow Rate	9
Nozzle Exit Area	6
Burn Time	4
Nozzle Area Ratio	4
Exhaust Velocity	4
Aktionszeit	3
Nozzle Throat Area	3
Chamber Mass	3
Äussere Klemmung	3
Specific Mass Flow Rate	3
Chamber Length	3
Fuel Mass Flow Rate	3
Abbrandgeschwindigkeit, Chamber Mixture Ratio, Conceptual Design End, Development Start, Coolant Mass Flow Rate, Program/Project Definition Start, Program/Project Definition End, Production Start, Production End, Prelimenary Design Start, Prelimenary Design End, Oxidizer Mass Flow Rate, Nozzle Throat Diameter, Zündverzugszeit, Van den Kerkhove-Faktor, Thrust Sea Level, Nozzle Exit Pressure, Nozzle Exit Diameter, Last Flight, Ignition Test/First Hot Firing, First Full Thrust Test, First Full Thrust & Full Burn Time Test, First Flight Test, Engine Width, Thrust Coefficient, Qualification Test End, Nozzle Mass, Engine Length, Characteristic Velocity, Abbrandoberfläche	2
Adiabatenexponent, Chamber Fuel Inlet Pressure Minimum, Chamber Inlet Pressure, Chamber Inlet Pressure Minimum, Chamber Inlet Pressure Maximum, spezielle Gaskonstante, halber Öffnungswinkel, Webb-Thickness, Tube Mass, Treibstoffdichte, Total Mass, Thrust Vacuum Min, Thrust Vacuum Max, Thrust Vacuum, Thrust Minimum, Thrust Maximum, PCCS Mass, Nozzle Throat Velocity, Nozzle Throat Pressure, Nozzle Length, Nozzle Expansion Ratio, Mixture Ratio Minimum, Mixture Ratio Maximum, Manifold Mass, Impulse Bit Minimum, Heat Exchanger Mass, Final Documentation/Report, Engine Height, Engine Diameter, Thrust Coefficient Vacuum, Thrust Coefficient Sea Level, Thrust, Specific Power, Specific Impulse Vacuum, Specific Impulse Sea Level, Specific Impulse, Nozzle Area Ratio End, Dry Mass, Dichte der Abgase am Düsenende, Controller Mass, Conceptual Design Start, Characteristic Length, Chamber Wall Temperature Maximum, Chamber Oxidizer Inlet Pressure Minimum, Chamber Oxidizer Inlet Pressure Maximum, Chamber Oxidizer Inlet Pressure, Chamber Fuel Inlet Temperature, Chamber Fuel Inlet Pressure Maximum, Chamber Combustion Temperature, Chamber Diameter, Chamber Fuel Inlet Pressure, Burn Time Maximum	1
Accel. Elect. Transparancy, Accel. Grid Holes Diam., Accel. Grid Holes Diameter, Acceleration Elect. Transparancy, Acceleration Grid Current, Acceleration Grid Holes Diameter, Acceleration Power Losses, Beam Current, Beam Current Maximum, Burn Time Per Mission, Burn Time Per Mission, Burnout Mass, Cathode Mass Flow, Chamber Oxidizer Inlet Temperature, Convergent Chamber Mass, Coolant Flow Rate, Current Density, Current Density Max, Current Density Min, Cycle Life, Cylindrical Chamber Mass, Deflection Maximum, Depth of Dishing, Developer, Discharge Current, Discharge Voltage, Divergent Chamber Mass, Droplet Diameter, Efficiency, Electric Power Efficiency, Electrical Power Input, Electrical Power Input Max, Energy Consumption In Pulse, Engine Regulation Range (Lower Limit), Engine Regulation Range (Mixture Ratio), Engine Regulation Range (Upper Limit), Expansion Area, Expansion Ratio, Feed System Mass, Full Power Level, Generator Frequency, Gimbal Angle Maximum, Gimbaling, Heater Current, Heater Voltage, Interelectrod Spacing, Ion Cost, Ion Cost Maximum, Ion Cost Minimum, Ionising Power, Jacket Mass, Lifetime, Main Designer, Manufacturer, Neutralizer Power, Number Of Restarts, PCU Height, PCU Length, PCU Width, PM Value, Poisson Ratio, Proof Pressure, Propellant Utilisation, Propellant Utilisation Maximum, Propellant Utilisation Minimum, Pulse Duration, Pulse Impulse, Response Time,10 % Thrust, Response Time,90 % Thrust, Screen Elect. Transparancy, Screen Grid Holes Diameter, Screen Grid Holes Width, Service Lifetime, Specific Discharge Energy, Start Heating-Up Time, Starting Heaters Power, Throttleability, Throttling Maximum, Throttling Minimum, Thruster Manufacturer, Time Off, Time On, Total Efficiency, Total Eqv.A Flow, U+, U-, Wet Mass, Manufacturer	0

Tab. 6-4 Anzahl Regeln pro Attribut für den Objekttyp `Engine`

Problem der Modellierungskonsistenz

In der Regelbasis sind zur Zeit Formeln aus verschiedenen Modellierungsansätzen und reine Schätzformeln erfasst. Diese werden von der Problemlösungskomponente ungeachtet ihrer Genauigkeit und Modellzugehörigkeit benutzt. Dies kann dazu führen, dass für eine Lösung Formeln und Regeln aus verschiedenen Modellierungen vermischt werden.

Eine Möglichkeit dies zu verhindern wäre eine strikte Trennung der Regeln. Jeder Modellierungsansatz würde zu einem eigenen Regelsatz führen, welcher wiederum völlig unabhängig von den anderen Regelsätzen benutzt werden kann. Der Benutzer kann nun festlegen, welche Regelsätze er benutzen will. Dadurch besteht die Möglichkeit sich auf die Verwendung bestimmter Klassen von Regeln zu beschränken und z.B. nur Regeln aus einem bestimmten Modell anzuwenden. Bei der Auswahl könnte man dem Benutzer per Programm verbieten unvereinbare Modelle zu kombinieren. Als zusätzlichen Effekt würde eine Beschränkung der Regelanzahl auch eine Beschleunigung der Lösungsfindung bewirken.

Eine andere flexiblere Möglichkeit wäre die Trennung unvereinbarer Modellierungen über den Einbau einer zusätzlichen Regel in die Voraussetzungsprüfung jeder Regel. Diese Regel würde anhand einer zusätzlichen Wissensbasis abprüfen, ob die zuvor angewandte Regel mit der aktuellen vereinbar ist oder nicht. Diese Überprüfung könnte ebenfalls auf Klassenzugehörigkeit erfolgen. Das hätte zur Folge, dass die erste angewandte Regel bestimmt, in welchem Modellierungsansatz sich der nachfolgende Lösungsweg bewegen wird.

Eine Kombination beider Möglichkeiten ist ebenfalls denkbar um die Vorteile beider Wege zu vereinen. Eine Aufteilung der Regeln in Klassen (z.B. gruppiert nach Modellierungen) welche dann, je nach Bedarf eingeschlossen oder ausgenommen werden. Zusätzlich eine Überprüfung der Anwendbarkeit einer Regel auch im Hinblick auf Modellierungskonsistenz. Da die Problemlösungskomponente immer alle möglichen Lösungen ermittelt, würde immer versucht werden alle zugeschalteten Modellierungen zu benutzen und konsistente Lösungen zu finden.

Problem der unerwünschten Rekursion

Dadurch, dass vor der eigentlichen Anwendung der Regel alle benötigten Attribute über Regeln beschafft werden, kann es zu unerwünschten Rekursionen kommen. Ein mögliches Problem ist die Benutzung von Regeln, die das gesuchte Attribut als Voraussetzung benötigen. Dadurch kann es passieren, dass bei der Beschaffung der benötigten Attribute die Regel eingesetzt wird, die den ursprünglichen Aufruf gestartet hat. Wenn dies passiert, werden die benötigten Attribute über Regeln beschafft usw. Dieser Fall tritt besonders dann auf, wenn eine Formel mit n Variablen für jede einzelne Variable aufgelöst wurde und somit n Regeln entstehen, die je eine Variable der Formel in Abhängigkeit von allen anderen liefern.

In Abb. 6-3 ist beispielhaft eine solche Situation dargestellt. Dieser Fall kann nicht nur mit der Regel, welche das gesuchte Attribut beschaffen soll passieren, sondern auch innerhalb des Suchbaums an einer beliebigen Stelle auftreten. Jede Regel kann, wenn keine Gegenmaßnahmen getroffen werden, ungewollt bei der Beschaffung der benötigten Attribute herangezogen werden.

Um diesen Fall zu verhindern, wurden zwei Bedingungen für die Anwendung einer Regel definiert:

- Jede Regel sollte innerhalb einer Resolution von einer anderen Regel nur einmal aufgerufen werden
- Regeln, die das gerade gesuchte Attribut als Bedingung benötigen, sollen nicht angewandt werden

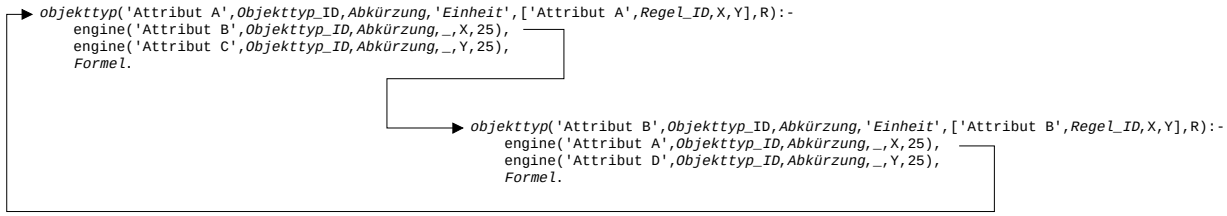


Abb. 6-3 Beispiel für unerwünschte Rekursion

Bei einem ersten Versuch war die erste Bedingung noch so formuliert, das jede Regel innerhalb der Resolution nur einmal angewendet werden konnte. Dies führte jedoch dazu, das in Fällen, in denen eine bestimmte Regel von verschiedenen Regeln als Voraussetzung benötigt wurde, keine Lösung mehr gefunden wurde. Daher wurde als zusätzliche Bedingung eingeführt, dass eine Regel von verschiedenen Regeln nur einmal aufgerufen werden kann.

Um das Einhalten dieser Bedingungen sicherzustellen, wurden die PROLOG Klauseln `not_used` und `not_wanted` programmiert. Die Klausel `hyp_asks_prolog` fügt mit dem PROLOG Kommando `asserta` eine zusätzliche PROLOG Klausel des Typs `wanted(`*`gesuchtes Attribut`*`)` ein. In jeder Regel wird nun vor Aufruf der Regeln, welche die benötigten Attribute beschaffen sollen durch Aufruf der Klausel

```
not_wanted(gesuchtes Attribut 1, gesuchtes Attribut 2, …, gesuchtes Attribut n)
```

überprüft, ob die von dieser Regel benötigten Attribute nicht das gerade gesuchte Attribut ist. Wie in der nachfolgenden Auflistung von `not_wanted/1` zu sehen ist, wird hierzu zunächst die Variable Y mit der aktuell gesuchten Attributbezeichnung instanziiert und danach mit der Liste X, die die benötigten Attribute der gerade anzuwendenden Regel enthält, verglichen. Findet sich eine Übereinstimmung schlägt `not_wanted` fehl und die aktuelle Regel wird nicht angewendet.

```
not_wanted(X):- wanted(Y), not member(Y,X).
```

Als erstes Subziel jeder Regel steht der Aufruf `not_used(Regel_ID,R.)`. Die Regel_ID wird als Zahlenwert übergeben und entspricht der Identifikationsnummer, mit der diese Regel in der AIM Datenbank gespeichert ist. Die Variable R wird von der diese Regel aufrufenden PROLOG Klausel übergeben und enthält die Identifikationsnummer der aufrufenden Regel. Wenn die Überprüfung ergibt, das die Regel angewendet werden kann, wird im nächsten Subziel mit der Klausel `assertz` eine Klausel der Form `used(`*`Regel_ID`*`,R)` eine zusätzliche PROLOG Klausel eingefügt. Die nachfolgend aufgeführte Klausel `not_used` überprüft, ob eine bestimmte Regel mit der `Regel_ID X` bereits von einer anderen Regel mit der `Regel_ID Y` aufgerufen wurde oder umgekehrt die Regel Y von der Regel X. Wenn beides nicht der Fall ist, kann mit der Verarbeitung der Regeln fortgefahren werden. Wird jedoch eine Benutzung festgestellt, dann wird die aktuelle Regel nicht ausgeführt, sondern nach anderen passenden Regeln gesucht.

```
not_used(X,Y):- not used(X,Y), not used(Y,X).
```

Alle zusätzlich eingefügten Klauseln müssen vor Beginn einer neuen Abfrage entfernt werden, da sonst nicht alle Lösungen gefunden werden können. Dies geschieht im Prädikat `hyp_asks_prolog` durch Aufruf des `retractall` Befehls:

```
retractall(used(A1,A2)),
retractall(wanted(C1)),
```

Problem der offenen Cursor

Beim Zusammenspiel zwischen PROLOG und ORACLE ergab sich noch ein ganz spezielles Problem. Das Ergebnis einer Datenbankabfrage, also eines SQL-Befehls, wird auf dem Datenbankserver in einem speziellen Speicherbereich abgelegt. Dieser wird als Cursor bezeichnet. Von dort können die einzelnen Zeilen des Resultats zur Client Anwendung übertragen werden. Dies kann einzeln oder in Blöcken geschehen. Normalerweise wird zwischen dem RDBMS und der Client-Anwendung nur ein Cursor benutzt und wenn ein neues SQL-Kommando Ergebnisse produziert, werden die alten Ergebnisse überschrieben. Will man von einer Client-Anwendung aus die Ergebnisse mehrerer SQL-Kommandos verknüpfen, dann werden mehrere Cursor geöffnet und die Ergebnisse, die sich ja nun in verschiedenen Cursorn befinden, können parallel abgefragt und verknüpft werden.

Da PROLOG mit Backtracking arbeitet und daher gezwungen ist auf alte Ergebnisse zurückzugreifen, wird für jedes SQL-Kommando ein eigener Cursor angelegt. Da sowohl PROLOG als auch das ORACLE RDBMS in der Anzahl der gleichzeitig geöffneten Cursor beschränkt sind, kam es bei sehr großen Suchtiefen zu einer Fehlermeldung, dass zu viele offene Cursor bestehen. Die Lösungssuche von PROLOG wurde abgebrochen und das PROLOG Programm beendet. Um dies zu vermeiden bieten sich zwei Lösungen an. Ein Abbrechen der Lösungssuche bei sehr tief verschachtelten Lösungen kann toleriert werden. Da aber die Suche nach weiteren Lösungen danach fortgesetzt werden soll, würde es reichen zu verhindern, dass das PROLOG Programm bei einer Fehlermeldung beendet wird. In `hyp_asks_prolog` befassen sich zwei Prädikate mit diesem Problem:

```
db_reset_cursors, db_flag(show_db_error,Old,off),
```

Zunächst werden bei einer neuen Lösungssuche alle eventuell noch offenen Cursor mit `db_reset_cursors` geschlossen. Dann wird mit `db_flag/3` das Anzeigen einer Fehlermeldung in Bezug auf das Datenbankinterface unterbunden. Dies hat zur Folge, dass der aktuelle Suchpfad bei datenbankbezogenen Fehlern mit einem "silent failure" beendet wird, aber alternative Pfade weiterhin nach Lösungen abgesucht werden.

Eine weitere Möglichkeit besteht darin die Anzahl der SQL-Abfragen zu verringern. Da während der Lösungssuche in der Regel nur eine eng begrenzte Anzahl Attribute aus der Datenbank abzufragen ist, kann durch programminternes "Caching" der Ergebnisse ein erneutes Absenden einer bereits ausgeführten SQL-Abfrage vermieden werden. Hierzu wird das Ergebnis einer Datenbankabfrage bestehend aus Attributbezeichnung, Zahlenwert, Einheit und Lösungsweg mit `asserta` am Anfang der PROLOG Klauseln als zusätzliche Klausel eingefügt. Wird jetzt der Attributwert benötigt, wird zuerst die Klausel angewendet, die im Programm zuerst aufgelistet ist. Danach muss vermieden werden, das die Regel mit dem SQL-Kommando erneut ausgeführt wird.

Problem der Geschwindigkeit

Wie bei allen KI Programmen und insbesondere bei wissensbasierten Systemen ist die Programmgeschwindigkeit abhängig vom Umfang des Suchraums. Im Fall des AIM Expertensystems wird der Umfang des Suchraums stark von der Anzahl der Regeln beeinflusst. Zusätzlich kann die Dauer der Lösungssuche noch durch die Anzahl der gefundenen Lösungen negativ beeinflusst werden.

Bei den Tests, die im Rahmen der Entwicklung durchgeführt wurden, waren Antwortzeiten von 1 Sekunde bis zu mehreren Minuten für die erste Lösung die Regel. Dies hängt aber stark davon ab, wieviel über das jeweilige Objekt bekannt ist. Sind viele Attributwerte als Fakten in der ORACLE Datenbank gespeichert, dann führt meistens eine der ersten Regeln direkt zu einer Lösung. Wird allerdings nach allen Lösungen gesucht, dann bewirkt das Vorhandensein vieler Attribute in der Datenbank, dass nahezu alle Regeln angewendet werden können, da nahezu alle Voraussetzungen für die Regeln zutreffen. Dadurch wird eine sehr große Anzahl an Lösungen gefunden und die vollständige Abarbeitung dauert entsprechend lange.

Abb. 6-4 zeigt das Ergebnis zweier Tests, die für den Objekttyp `stage` durchgeführt wurden. Es wurde eine Teststufe eingefügt, für die sämtliche möglichen Attribute in der Datenbank gespeichert waren. Für den Test wurde das Attribut "Net Mass" gelöscht und dann als Frage an das Expertensystem übergeben. Die Suche wurde zweimal ausgeführt, um Schwankungen durch unterschiedliche Auslastung des Datenbankservers auszuschalten und möglichst nur die Performance des Regelteils des Expertensystems zu testen. Es wurde nach allen Lösungen gefragt. Insgesamt wurden 56 Lösungen gefunden. Jede Lösung wurde auf einem unterschiedlichen Lösungsweg ermittelt. Für alle 56 Lösungen brauchte das Expertensystem 17:24 min im ersten und 17:18 min im zweiten Fall.

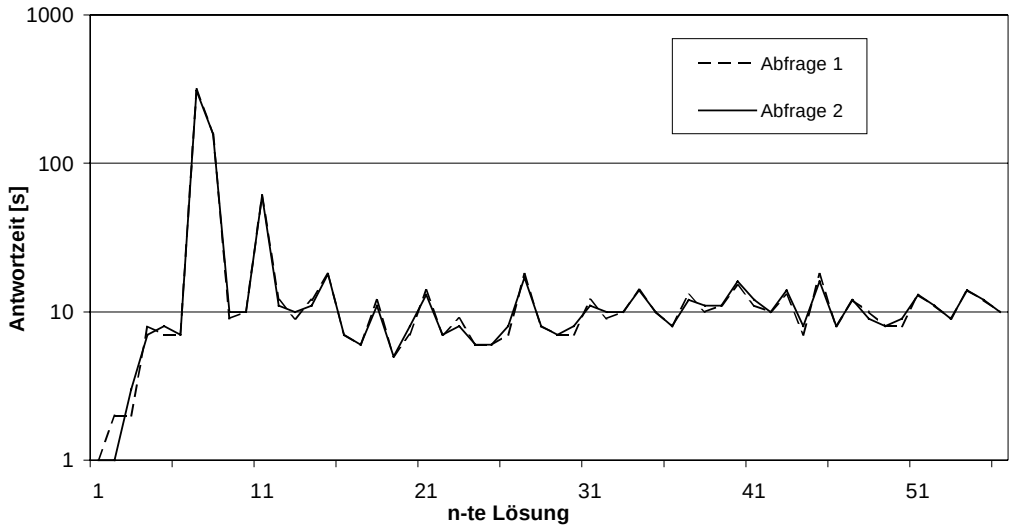


Abb. 6-4 Antwortzeiten des Expertensystems

Der Verlauf in der Abbildung zeigt, welche Zeitspanne zwischen jeweils zwei Lösungen vergeht. Die erste Lösung wurde nach 1 Sekunde gefunden. Danach dauerte das Auffinden weiterer Lösungen immer längere Zeitspannen bis das Maximum bei der 7. Lösung erreicht war. Danach pendelt sich die Antwortzeit bei 10 Sekunden ein.

Wiederholt man die Tests, ohne die Fehlermeldung der offenen Cursor zu unterdrücken, so stellt man fest, das der Abbruch bei der Suche nach der 7. Lösung auftritt. Dies lässt vermuten, dass bei der Untersuchung von Alternativlösungen mehrere "silent failures" auftreten, die natürlich die Suche nach der dann gefundenen 7. Lösung verzögern. Außerdem muss man daraus folgern, das nicht alle Lösungen zwischen der 6. und 7. Lösung gefunden werden.

Pseudo Attribute

Für einige Regeln werden Attributwerte benötigt, deren Attributbezeichnung nicht in der Datenbank vorkommt. Einige dieser Attribute, die nicht explizit in der Datenbank vorhanden sind, können aber durch spezielle SQL-Abfragen beschafft werden. Zum Beispiel ist die Anzahl der Triebwerke einer Stufe implizit in der Datenbank gespeichert und kann durch eine eigene Regel, die eine spezielle SQL-Abfrage enthält aus der Datenbank abgefragt werden. Hierfür wird eine Regel des Objekttyps `stage` programmiert, die wie folgt aufgebaut ist:

```
stage('Number Of Engines',Stage_ID,NOE,'-',['Number Of Engines',8000],R):-
   not_used(433,R),
   asserta(used(433,R)),
   db_sql_select(['SELECT SUM(Stage_EngineAmount) FROM Stage_Engine
                   WHERE Stage_ID = ', Stage_ID ], NOE),
   not NOE=[].
```

Diese Regel summiert die Einträge in der Spalte `Stage_EngineAmount` der Tabelle `Stage_Engine` auf und ermittelt so die Anzahl der Triebwerke in der betreffenden Stufe.

Zusätzlich gibt es noch Attributwerte, die gar nicht in der Datenbank gespeichert sind, aber von einigen Regeln benötigt werden. Es wurde auf die Speicherung in der Datenbank verzichtet, da diese Informationen unter normalen Umständen vom Benutzer nicht benötigt werden. Außerdem sind diese Informationen so exotisch, dass sie in der Literatur ebenfalls nicht angegeben werden und somit auch für die meisten Objekte nicht zur Verfügung stehen. Einige spezielle Regeln benötigen jedoch diese Informationen als Voraussetzung für ihre Anwendung. Daher werden sie über Regeln erzeugt und wie Pseudo-Attribute behandelt. Attribute dieser Art sind der Adiabatenexponent der Triebwerksabgase und der van-den-Kerkhove-Faktor.

Durch die Einführung von Pseudo-Attributen können solche Eigenschaften wie normale Attribute durch das PROLOG Programm behandelt werden. Dadurch sind keine umständlichen Programmierungen notwendig und es kann auch auf versteckte Informationen in der Datenbank, aber auch auf Informationen, die nicht in der Datenbank gespeichert sind, durch den Benutzer in der gewohnten einfachen Weise zugegriffen werden.

6.5 Erklärungskomponente

Wie bereits im Abschnitt 3.2 erläutert, dient die Erklärungskomponente zur Illustration des Lösungsweges um eine bessere Akzeptanz der gefundenen Lösung beim Benutzer zu erreichen. Die Erklärungskomponente des AIM Expertensystems ist ein HyperCard Programm welches seine Informationen zum Lösungsweg und zu den angewandten Regeln von der PROLOG Problemlösungskomponente und der ORACLE Datenbank bezieht. Basierend auf der von PROLOG angebotenen Lösung kann die Reihenfolge der angewendeten Regeln extrahiert werden. Die Übergabe der gefundenen Lösung(en) an HyperCard geschieht durch die Datei `prol_to_hyp`, welche bereits in Kap. 6.1 beschrieben wurde. Jede Zeile der Datei stellt eine gefundenen Lösung dar. Der in Abb. 6-5 gezeigte Ausschnitt aus der Datei zeigt sechs Lösungen für die Brennschlussmasse (Burn-Out Mass) einer Stufe. Alle Lösungen sind mit eckigen Klammern (den PROLOG Listenzeichen) eingeschlossen. Zu Beginn jeder Zeile steht der gesuchte Zahlenwert und soweit sinnvoll die Einheit. Jede Zeile stellt eine PROLOG Liste dar, welche wiederum aus Listen besteht. Diese Listen stellen die Lösungswege für die Subziele zur Erfüllung des Oberziels (der gesuchten Eigenschaft) dar. Jedes Subziel wird durch eine gesuchte Eigenschaft repräsentiert, die zur Erfüllung des Oberziels notwendig ist. Der Eigenschaft folgt durch Komma getrennt eine weitere Liste mit Subzielen, die wiederum zur Erfüllung des gerade aktuellen Subziels notwendig sind, usw. Die Zahlenwerte bei den Subzielen sind die `Regel_IDs` der verwendeten Regeln.

```
[10, kg, [Burn-Out Mass, 0, 9008]]
[10, kg, [Burn-Out Mass, 445, [Net Mass, 0, 9008]]]
[0, kg, [Burn-Out Mass, 445, [Net Mass, 469, [Lift-Off Mass, 0, 9008], [Usable Propellant
Mass, 0, 9008]]]]
[60, kg, [Burn-Out Mass, 445, [Net Mass, 469, [Lift-Off Mass, 0, 9008], [Usable Propellant
Mass, 500, [Recovery System Mass, 0, 9008], [Propulsion System Mass, 0, 9008], [Guidance and
Control Mass, 0, 9008], [Structure Mass, 0, 9008], [Usable Extra Propellant Mass, 0, 9008],
[Lift-Off Mass, 0, 9008], [Propellant Residuals Mass, 0, 9008]]]]]
[0, kg, [Burn-Out Mass, 445, [Net Mass, 469, [Lift-Off Mass, 0, 9008], [Usable Propellant
Mass, 500, [Recovery System Mass, 0, 9008], [Propulsion System Mass, 0, 9008], [Guidance and
Control Mass, 0, 9008], [Structure Mass, 0, 9008], [Usable Extra Propellant Mass, 0, 9008],
[Lift-Off Mass, 0, 9008], [Propellant Residuals Mass, 474, [Structure Mass, 0, 9008],
[Recovery System Mass, 0, 9008], [Propulsion System Mass, 0, 9008], [Usable Extra Propellant
Mass, 0, 9008], [Usable Propellant Mass, 0, 9008], [Guidance and Control Mass, 0, 9008],
[Lift-Off Mass, 0, 9008]]]]]]
[60, kg, [Burn-Out Mass, 445, [Net Mass, 469, [Lift-Off Mass, 0, 9008], [Usable Propellant
Mass, 500, [Recovery System Mass, 0, 9008], [Propulsion System Mass, 0, 9008], [Guidance and
Control Mass, 0, 9008], [Structure Mass, 0, 9008], [Usable Extra Propellant Mass, 0, 9008],
[Lift-Off Mass, 0, 9008], [Propellant Residuals Mass, 474, [Structure Mass, 0, 9008],
[Recovery System Mass, 0, 9008], [Propulsion System Mass, 0, 9008], [Usable Extra Propellant
Mass, 0, 9008], [Usable Propellant Mass, 0, 9008], [Guidance and Control Mass, 0, 9008],
[Lift-Off Mass, 456, [Structure Mass, 0, 9008], [Recovery System Mass, 0, 9008], [Propulsion
System Mass, 0, 9008], [Usable Extra Propellant Mass, 0, 9008], [Usable Propellant Mass, 0,
9008], [Guidance and Control Mass, 0, 9008], [Propellant Residuals Mass, 0, 9008]]]]]]]
```

Abb. 6-5 Ausschnitt aus der Übergabedatei `prol_to_hyp`

Das HyperCard Benutzerinterface extrahiert zunächst alle Lösungen für das Oberziel aus der `prol_to_hyp` Datei und stellt sie tabellarisch mit den entsprechenden Einheiten, wie in Abb. 6-6 gezeigt, dar. Der Benutzer erhält so einen Überblick über alle gefundenen Lösungen. Wenn es sich um einen konsistenten Datensatz handelt, dürfen die gefundenen Lösungen nur geringfügig voneinander abweichen. Bei größeren Abweichungen liegt die Vermutung nahe, das einzelne Werte des Datensatzes inkorrekt sind. Die entsprechende Lösung wird vom Benutzerinterface markiert und die an der Lösung beteiligten Fakten aus der AIM Datenbank als potentiell falsch eingestuft. Der Benutzer kann sich diese Fakten über ein Kontextmenu (Apple-Taste + Mausklick auf die markierte Lösung) auflisten lassen. Über dasselbe Kontextmenu erhält er die übersichtliche Darstellung des jeweiligen Lösungsweges.

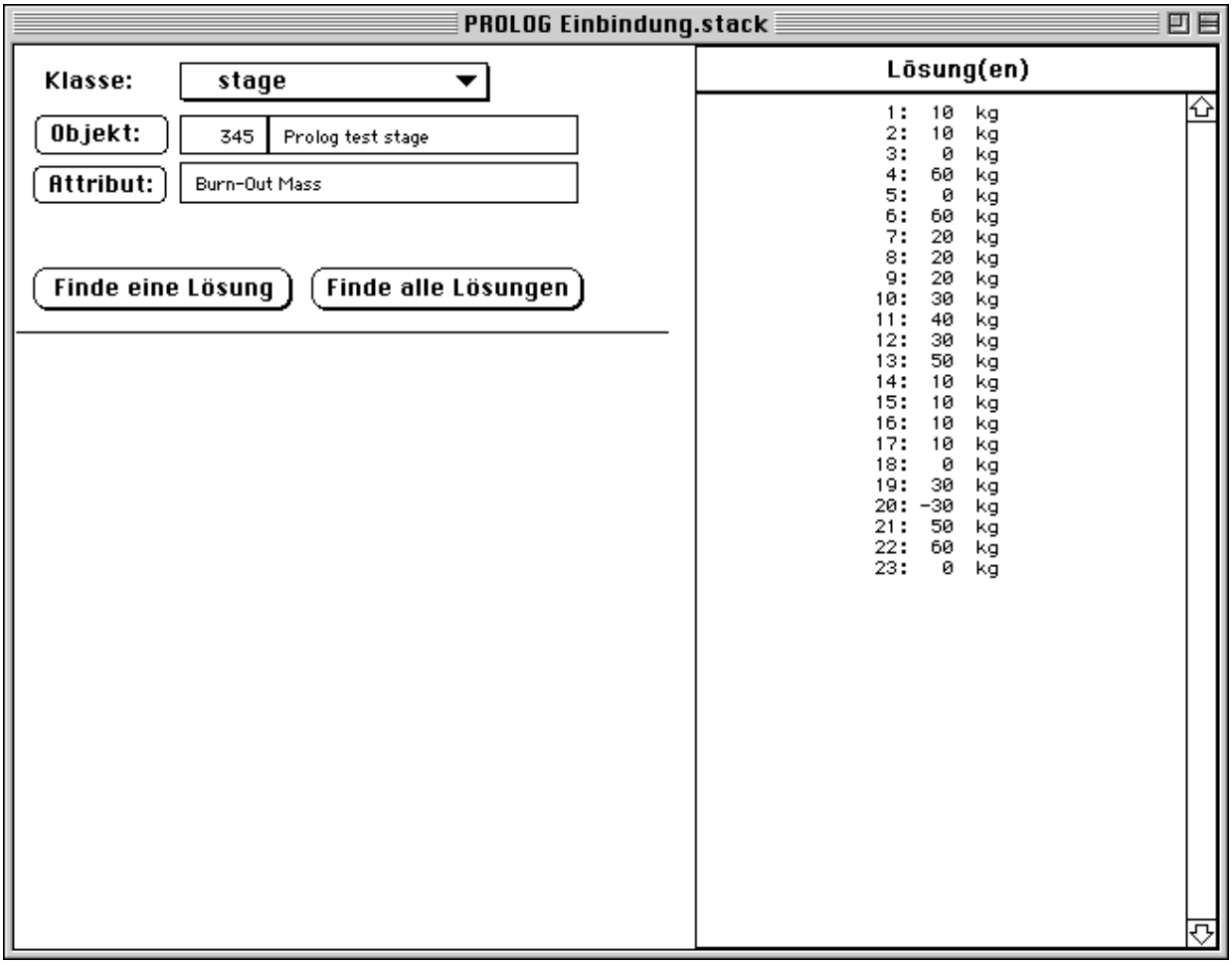


Abb. 6-6 Liste der Antworten auf eine Anfrage

Die Erklärungskomponente formatiert der Lösungsweg aus der `prol_to_hyp` Datei, so dass eine übersichtliche Darstellung erreicht wird. In Abb. 6-7 ist ein Lösungsweg in der übersichtlichen Darstellung gezeigt. In der ersten Zeile steht das gesuchte Attribut und die Regel(n) mit denen es erzeugt wurde. Darunter in der nächsten Zeile, etwas eingerückt, die von der Regel benötigten Attribute mit den Regeln, die wiederum diese benötigten Attribute erzeugt haben. Dadurch entsteht eine Art Baumstruktur, an deren Spitze die gefundene Lösung steht und an deren anderen Enden jeweils Zugriffe auf das Faktenwissen der AIM Datenbank stehen. Dazwischen liegt eine beliebig lange Kette von Regeln, die alle aufeinander aufbauen.

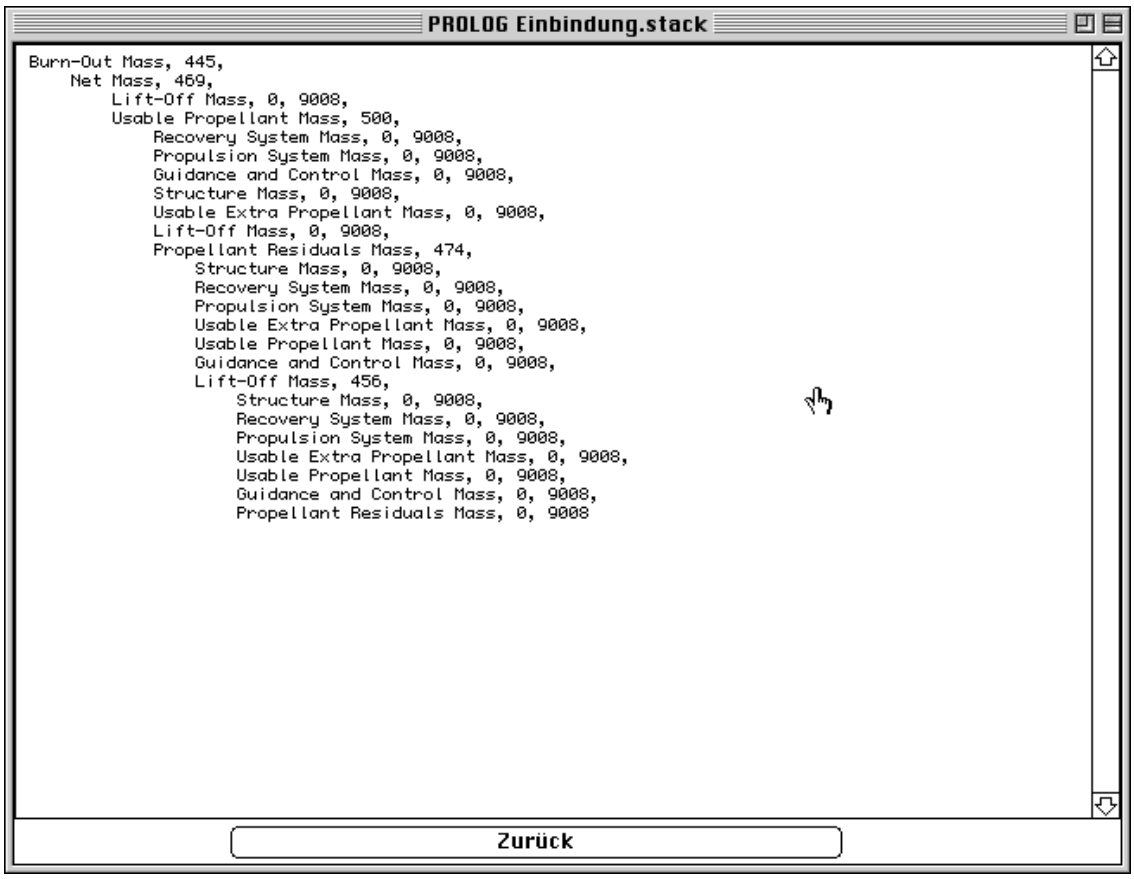


Abb. 6-7 Übersichtliche Darstellung des Lösungswegs

In der übersichtlichen Darstellung können durch Mausklick auf eine `Regel_ID` zusätzliche Informationen zu dieser Regel abgerufen werden. In einem eigenen Fenster (s.a. Abb. 6-8) werden Regel_ID, Regelname, Konfidenzfaktor und eventuelle Kommentare aus der AIM Datenbank abgefragt und angezeigt. Zusätzlich erhält der Benutzer eine Liste mit den von dieser Regel benötigten Attributen sowie das gelieferte Attribut, jeweils ergänzt um den Objekttyp (z.B. `Stage.Propellant Mass` oder `Launcher.Lift-Off Mass`). Diese Ergänzung ist für die Unterscheidung der Attribute notwendig, da einige Attribute bei verschiedenen Objekttypen den gleichen Namen besitzen. Diese Informationen werden online aus den Tabellen `Rule`, `Rule_Result`, und `Rule_Condition` per SQL-Abfrage zusammengestellt. Daher müssen für jede neue PROLOG Regel die entsprechenden Einträge in den AIM Tabellen vorgenommen werden.

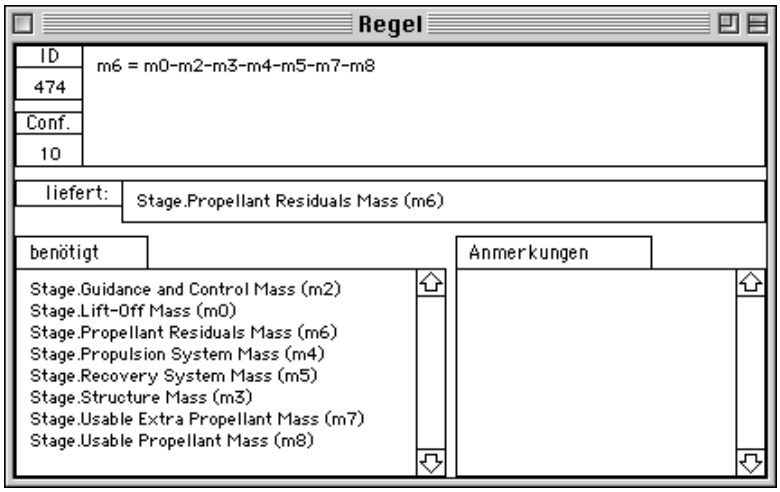


Abb. 6-8 Detailinformationen zu einer Regel

In der derzeitigen Version wird nur eine kurze Erläuterung der Regel oder die benutzte Formel dargestellt. Für einen Einsatz als Tutorial, oder als Erklärung für Raumfahrtlaien ist dies natürlich zuwenig. In späteren Versionen ist folglich auf eine ausführliche Regelerklärung zu achten. Dabei muss die theoretische Grundlage jeder Regel auch für den Laien nachvollziehbar sein. Auch die Voraussetzungen für die Anwendung der Regel bestehen zur Zeit nur darin, dass die in der Regel (i.a. Formel) verwendeten Attribute vorhanden sein müssen. Auch hier muss eine zusätzliche Erklärung erfolgen unter welchen Voraussetzungen diese Regel angewendet werden kann, bzw. wann von einer Anwendung der Regel abgesehen werden muss. Es würde sich zusätzlich anbieten an dieser Stelle der Erklärungskomponente auf die verschiedenen Modellierungen einzugehen und die Annahmen der jeweils verwendeten Modellierung detailliert darzustellen.

7 Zugriff auf die Informationen

In diesem Teil der Arbeit werden die unterschiedlichen Zugriffsarten auf Datenbanken und Informationssysteme kurz dargestellt und verglichen. Zusätzlich zu reinen Abfragewerkzeugen wird kurz auf die nicht weniger wichtige Datenmanipulationssoftware eingegangen. Abschließend werden die in AIM verwirklichten Zugriffsmethoden vorgestellt.

Unter Berücksichtigung der Einsatzmöglichkeiten von KI-Methoden bei der Informationssuche kann man folgende Arten von Benutzerschnittstellen unterscheiden:

- Benutzerschnittstellen bei denen die "Intelligenz" beim Benutzer liegt. Der Benutzer benötigt Erfahrung im Umgang mit Informationssystemen, sowie ein gewisses Grundwissen über das Fachgebiet aus dem die abzurufenden Informationen stammen. Außerdem muss er im Umgang mit Computern Erfahrungen gesammelt haben.
- Benutzerschnittstellen, bei denen ein Teil der Intelligenz in der Schnittstellensoftware bzw. im Informationssystem liegt und ein gewisses Teilwissen beim Benutzer vorausgesetzt wird.
- Benutzerschnittstellen bei denen der Benutzer nur sehr geringe oder gar keine Vorkenntnisse benötigt und eine intensive Anwendung von KI-Methoden erfolgt, um dem Benutzer die bestmögliche Antwort auf seine Anfragen zu geben. Diese Systeme verhalten sich extrem fehlertolerant.

Die ersten beiden Arten sind im AIM Informationssystem vorhanden und werden im folgenden näher erläutert. Der letzte Punkt steht für einen nicht existenten Idealfall, der zwar zur Zeit fast unerreichbar ist, auf den aber im folgenden Abschnitt kurz eingegangen werden soll, da er das Idealziel aller Benutzerschnittstellen darstellt. Aufgabe der Programmierer ist es, diesem Zustand möglichst nahe zu kommen. Hierzu sind nicht immer KI-Methoden nötig, da auch mit konventionellen Programmiertechniken eine Annäherung an den Idealzustand möglich ist. Jedoch erleichtert die Anwendung der Techniken aus dem Bereich der KI die Verwirklichung zum Teil erheblich.

7.1 Die ideale Informationssuche

Als Optimum an Benutzerfreundlichkeit wird allgemein ein System angesehen, welches ein natursprachliches Benutzerinterface besitzt. Man unterhält sich mit dem Computer in normalen Sätzen und formuliert seine Anfragen wie man einen menschlichen Experten fragen würde. Da es sich bei Informations- und Expertensystemen und gerade auch beim AIM System in der Regel um fachlich eng begrenzte Systeme handelt, kann der Wortschatz entsprechend eingeschränkt werden. Die Benutzerschnittstelle sollte dann in der Lage sein das Anliegen des Benutzers an Hand verschiedener Schlüsselworte zu erkennen oder im Zweifelsfall sinnvoll nachzufragen. Für das Erkennen der Benutzerbedürfnisse können auch alte gespeicherte Anfragen herangezogen werden, da sich der Großteil der an das Informationssystem gestellten Anfragen erfahrungsgemäss nicht groß unterscheidet. Dann benutzt das System seine Kenntnisse über den Aufbau der Datenbank, welche als Fakten und Regeln in einer Wissensbasis abgelegt sind, um die verlangten Informationen zu beschaffen. Zum Schluss müssen diese noch in eine Form gebracht werden, in der sie dem Benutzer am Vorteilhaftesten präsentiert werden können. Der Benutzer sollte in der Lage sein diesen Prozess gegebenenfalls zu steuern, so dass er die Informationen in der für ihn angenehmsten Art erhält.

Wie geschildert gliedert sich dieser Idealfall in drei unabhängige Phasen:

- Annahme der Benutzeranfrage und Übersetzung in eine für das System verständliche Form
- Beschaffung der notwendigen Informationen aus den dem System zur Verfügung stehenden Quellen (Datenbank, Regelwissen, etc.)
- Darstellung der gefundenen Informationen

Diese Phasen lassen sich auch bei den im folgenden geschilderten AIM Zugriffsmethoden erkennen. Obwohl noch weit von der Idealform entfernt sind einige der geschilderten Methoden bereits in der AIM Abfragesoftware verwirklicht.

7.2 Zugriffsmöglichkeiten auf das AIM Informations- und Expertensystem

Die vielfältigen Möglichkeiten des Zugriffs auf das AIM Informationssystem erstrecken sich von der einfachen Eingabe von SQL-Kommandos, bei der umfangreiches Wissen über den Aufbau der Datenbank unabdingbar ist, bis hin zu komfortablen grafischen Benutzerschnittstellen. Ein weiterer Vorteil ist die Möglichkeit über kommerzielle Standardsoftware auf den Datenbestand zuzugreifen. Dies wurde durch die Verwendung von allgemein gebräuchlicher Software erreicht, die verschiedenste Protokolle unterstützt. In Abb. 7-1 ist eine Übersicht über die beim Zugriff auf das Informationssystem verwendeten Programme und Protokolle dargestellt.

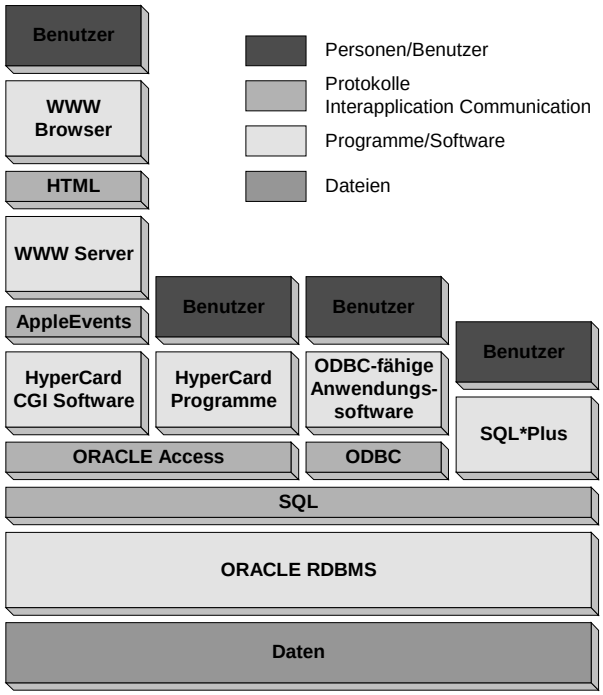


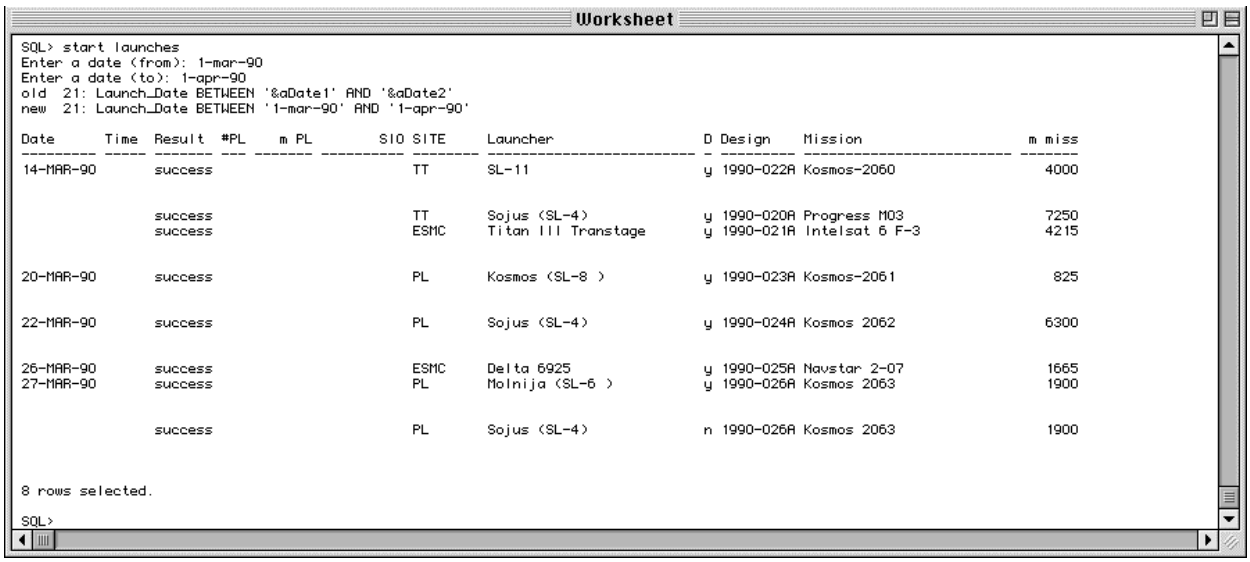
Abb. 7-1 AIM Zugriffsschichten

Zugriff über direkte Eingabe von SQL-Kommandos

Die direkte Eingabe von SQL-Kommandos stellt die flexibelste Möglichkeit der Datenabfrage dar. Allerdings muss der Benutzer sich sowohl mit der SQL Syntax als auch mit der Struktur der Datenbank auskennen. Er muss alle Tabellen- und Spaltennamen und die bestehenden Relationen kennen. Für die Eingabe von SQL-Befehlen im AIM Informationssystem wird die dem ORACLE Server beiliegende Software SQL*Plus benutzt. Dieses Programm stellt eine nicht-grafische zeilenorientierte Schnittstelle dar, in der unter einem Prompt Befehle eingegeben und die Ergebnisse angezeigt werden.

Das Ergebnis einer SQL-Abfrage an eine Datenbank sieht im einfachsten Fall wie in Abb. 7-2 gezeigt aus. Für das gezeigte Ergebnis wurden mehrere Tabellen miteinander verknüpft, da die gesuchten Informationen über mehrere Tabellen verteilt waren. Außerdem wurde bereits eine einfache Formatierung des Ergebnisses mit Hilfe von SQL-Kommandos vorgenommen.

Wenn eine Abfrage eine ganze Liste von Zeilen als Ergebnis liefert, dann ist die tabellarische Darstellung nahezu unvermeidbar. Wenn sich unter den darzustellenden Spalten jedoch mehrzeilige Informationen wie z.B. lange Texte oder Kommentare befinden, muss man entweder Informationen abschneiden oder zu Lasten der Übersichtlichkeit mehrzeilig umbrechen.



Worksheet

```
SQL> start launches
Enter a date (from): 1-mar-90
Enter a date (to): 1-apr-90
old  21: Launch_Date BETWEEN '&aDate1' AND '&aDate2'
new  21: Launch_Date BETWEEN '1-mar-90' AND '1-apr-90'

Date      Time  Result  #PL    m PL        SIO SITE     Launcher                 D Design    Mission                    m miss
--------- ----- ------- --- ------- ---------- -------- ------------------------ - --------- ------------------------ -------
14-MAR-90       success                        TT       SL-11                    y 1990-022A Kosmos-2060                4000


                success                        TT       Sojus (SL-4)             y 1990-020A Progress M03               7250
                success                        ESMC     Titan III Transtage      y 1990-021A Intelsat 6 F-3             4215


20-MAR-90       success                        PL       Kosmos (SL-8 )           y 1990-023A Kosmos-2061                 825


22-MAR-90       success                        PL       Sojus (SL-4)             y 1990-024A Kosmos 2062                6300


26-MAR-90       success                        ESMC     Delta 6925               y 1990-025A Navstar 2-07               1665
27-MAR-90       success                        PL       Molnija (SL-6 )          y 1990-026A Kosmos 2063                1900


                success                        PL       Sojus (SL-4)             n 1990-026A Kosmos 2063                1900


8 rows selected.

SQL>
```

Abb. 7-2 Tabellarische Darstellung des Ergebnisses eines SQL-Befehls (ORACLE SQL*Plus)

Die Anzahl der beteiligten Tabellen und die notwendigen Verknüpfungen für das in Abb. 7-2 gezeigte Ergebnis, gehen aus dem im folgenden aufgelisteten SQL*Plus Kommando hervor:

```
SET PAGESIZE 500
   SET LINESIZE 500
ACCEPT aDate1 PROMPT 'Enter a date (from): '
   ACCEPT aDate2 PROMPT 'Enter a date (to): '
COLUMN Launch_Date HEADING Date
   COLUMN Time HEADING Time FORMAT A5
   COLUMN Result HEADING Result FORMAT A7
   COLUMN PL HEADING #PL FORMAT 99
   COLUMN mPL HEADING "m PL" FORMAT 999999
   COLUMN Mission_LiftOffMass HEADING "m miss" Format 999999
   COLUMN Launch_StagesInOrbit Heading SiO FORMAT 90
   COLUMN Launchsite_Shortname HEADING Site FORMAT A5
   COLUMN Launcher_Name HEADING Launcher FORMAT A25 TRUNCATE
   COLUMN Launch_LauncherDefault HEADING D
   COLUMN Mission_Name HEADING Mission FORMAT A25 TRUNCATE
   COLUMN Mission_IntDesignation HEADING Design FORMAT A9
BREAK ON Launch_Date
SELECT
   Launch_Date,
   TO_CHAR(LAUNCH_TIME,'HH:MM') Time,
   Launch_Result Result,
   Launch_NoPayloads PL,
   Launch_MassPayloads mPL,
   Launch_StagesInOrbit SiO,
   Launchsite_Shortname Site,
   Launcher_Name,
   Launch_LauncherDefault def,
   Mission_IntDesignation,
   Mission_Name,
   Mission_LiftOffMass
   FROM
   Launch,
   Launch_Launcher,
   Launcher,
   Mission,
   Launchsite
   WHERE
   Launch_Date BETWEEN '&aDate1' AND '&aDate2'
   AND Launch_Launcher.Launcher_ID = Launcher.Launcher_ID
   AND Launch_Launcher.Launch_ID = Launch.Launch_ID
   AND Mission.Launch_ID = Launch.Launch_ID
   AND Launch_Launcher.Launchsite_ID = Launchsite.Launchsite_ID
   ORDER BY
   Launch_Date, Launcher_Name;
```

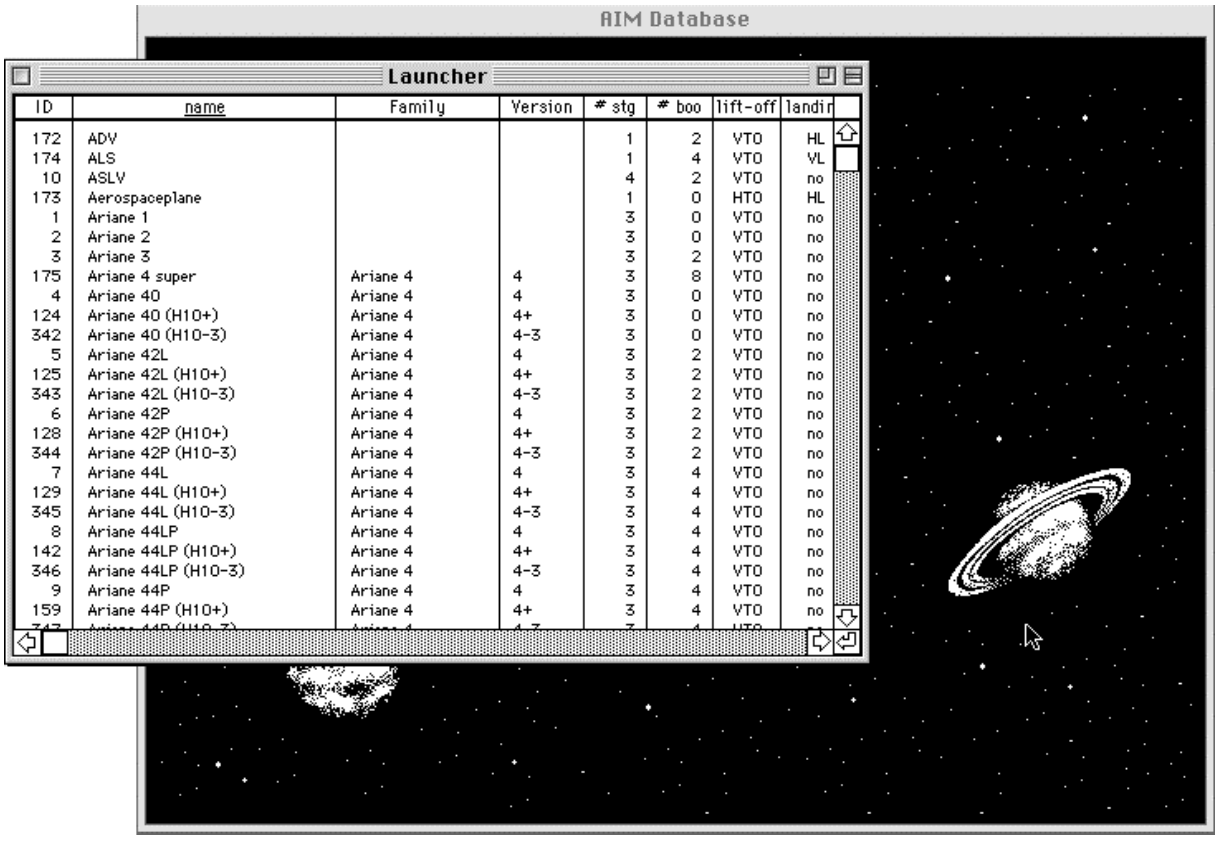
ODBC Zugriff

Es existieren zahlreiche kommerzielle Softwareprodukte, die über den ODBC Standard (ODBC = Open Database Connectivity) auf Datenbanken lokal oder im Netz zugreifen. So ist zum Beispiel Microsoft Access oder Microsoft Excel in der Lage über ODBC auf die ORACLE Datenbank zuzugreifen. Diese Programme stellen eine Steigerung gegenüber SQL*Plus dar, da unter einer grafischen Oberfläche SQL-Befehle sehr bequem zusammengestellt und an die Datenbank abgeschickt werden können. Allerdings ist hier auch schon der Trend erkennbar, dass mit steigender Benutzerfreundlichkeit die Flexibilität zurückgeht.

HyperCard Oberfläche

Für die Manipulation des Datenbestands sowie für die Eingabe neuer Daten wurden mehrere grafische Oberflächen unter HyperCard auf dem Apple Macintosh programmiert. Einige der Programme erlauben die Abfrage und Manipulation von Teilbereichen der Datenbank. So ist z.B. eine sehr bequeme Manipulation der in der AIM Datenbank gespeicherten Literaturstellen möglich. Eine sehr umfangreiche Sammlung von HyperCard Stapeln dient als Datenmanipulationssoftware für die gesamte Datenbank. Diese erlaubt das Einfügen, Löschen und Ändern von einzelnen Datensätzen oder mehreren Datensätzen zugleich.

Abb. 7-3 zeigt einen Ausschnitt dieser Manipulationssoftware für die Dateneingabe und Datenänderung des AIM Informationssystems. Da man bei Änderungen in der Regel eine Übersicht über den vorhandenen Datenbestand benötigt, wurde auch hier die tabellarische Darstellung der Informationen gewählt. Im Gegensatz zu Abb. 7-2 wird hier jedoch die tatsächliche Tabellenstruktur der AIM Datenbank wiedergegeben. Das heißt die Abb. 7-3 zeigt die Struktur (Spalten) der Tabelle `Launcher` so wie sie auch in der Datenbank vorhanden ist. Das erschwert zwar die Übersicht über die Verknüpfungen (Relationen) zwischen den Objekten (Tabellen), jedoch erhält der Benutzer einen Eindruck der tatsächlich in einer Tabelle vorhandenen Daten.



ID	name	Family	Version	# stg	# boo	lift-off	landir
172	ADV			1	2	VTO	HL
174	ALS			1	4	VTO	VL
10	ASLV			4	2	VTO	no
173	Aerospaceplane			1	0	HTO	HL
1	Ariane 1			3	0	VTO	no
2	Ariane 2			3	0	VTO	no
3	Ariane 3			3	2	VTO	no
175	Ariane 4 super	Ariane 4	4	3	8	VTO	no
4	Ariane 40	Ariane 4	4	3	0	VTO	no
124	Ariane 40 (H10+)	Ariane 4	4+	3	0	VTO	no
342	Ariane 40 (H10-3)	Ariane 4	4-3	3	0	VTO	no
5	Ariane 42L	Ariane 4	4	3	2	VTO	no
125	Ariane 42L (H10+)	Ariane 4	4+	3	2	VTO	no
343	Ariane 42L (H10-3)	Ariane 4	4-3	3	2	VTO	no
6	Ariane 42P	Ariane 4	4	3	2	VTO	no
128	Ariane 42P (H10+)	Ariane 4	4+	3	2	VTO	no
344	Ariane 42P (H10-3)	Ariane 4	4-3	3	2	VTO	no
7	Ariane 44L	Ariane 4	4	3	4	VTO	no
129	Ariane 44L (H10+)	Ariane 4	4+	3	4	VTO	no
345	Ariane 44L (H10-3)	Ariane 4	4-3	3	4	VTO	no
8	Ariane 44LP	Ariane 4	4	3	4	VTO	no
142	Ariane 44LP (H10+)	Ariane 4	4+	3	4	VTO	no
346	Ariane 44LP (H10-3)	Ariane 4	4-3	3	4	VTO	no
9	Ariane 44P	Ariane 4	4	3	4	VTO	no
159	Ariane 44P (H10+)	Ariane 4	4+	3	4	VTO	no

Abb. 7-3 Tabellarische Darstellung einer Tabelle/Objekt in der AIM Manipulationssoftware

Zusätzlich zur Datenmanipulationssoftware für alle Tabellen wurden mehrere kleinere Bearbeitungstools für Teilbereiche der AIM Datenbank in HyperCard entwickelt (z.B. für das Eintragen von Startdaten oder das Eintragen und Zuordnen von Bildern und Filmen). Zwei größere HyperCard Programme befassen sich mit der Manipulation und Abfrage der Literaturdaten. Dieser wichtige Teilbereich der AIM Datenbank erfüllt zwei wichtige Aufgaben. Zum Einen dienen die Einträge im Literaturteil als Referenzen für die in der Datenbank gespeicherten Zahlenwerte. Zum Anderen ist durch die Speicherung aller in der Fachgebietsbibliothek vorhandenen Bücher, Vortragspaper, Broschüren, Dissertationen, Diplomarbeiten und Studienarbeiten ein schneller Zugriff auf diese Literatur möglich. Durch die Zuordnung von Stichworten ist somit eine Suche nach verschiedensten Kriterien (wie z.B. Titel, Autor, Jahr, Stichwort, Organisation, etc.) möglich. In Abb. 7-4 ist die Suchmaske für die Literaturdatenbank zu sehen. Die gefundenen Literaturstellen werden als Liste oder einzeln in Karteikartenform (s.a. Abb. 7-5) angezeigt. Die eigentliche Literatur kann dann dem entsprechenden Ordner in der Fachgebietsbibliothek entnommen werden. Durch die gleichzeitige Verwendung der Literaturdaten als Quellennachweis für die gespeicherten Informationen kann es vorkommen, dass eingetragene Quellen nicht in der Fachgebietsbibliothek vorhanden sind, sondern nur zum Zweck des Quellennachweises für die AIM Datenbank eingetragen wurden.

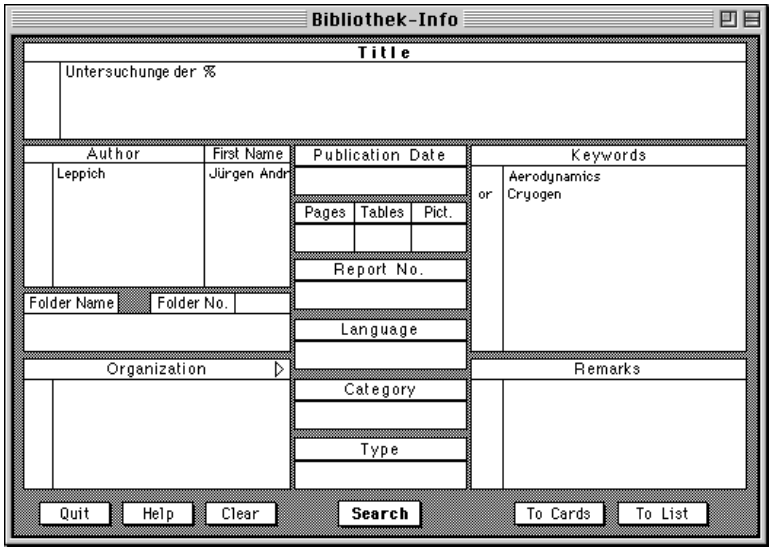


Abb. 7-4 Suchmaske der Bibliothekssoftware

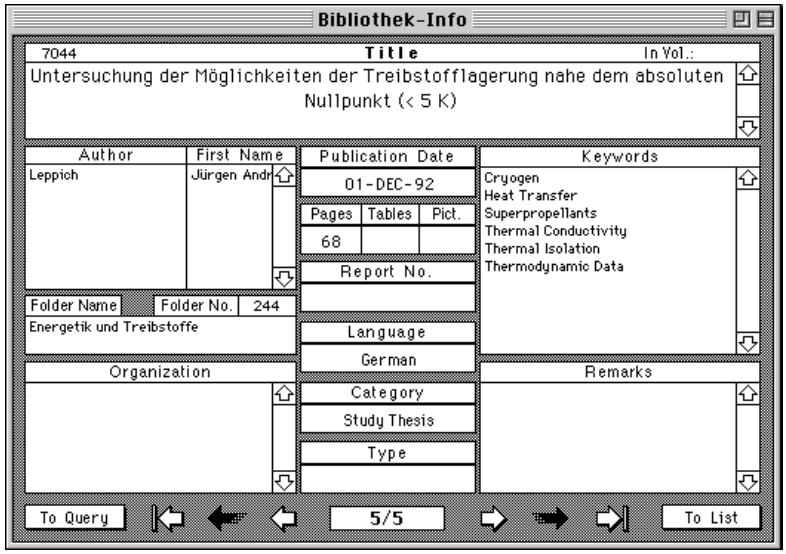


Abb. 7-5 Anzeige einer Literaturstelle in der Bibliothekssoftware

Zugriff über das Internet (WWW)

Bei Informationssystemen neuerer Prägung werden nicht mehr nur reine Zahlen und Fakten gespeichert, sondern vielmehr Bilder, Filme, Töne und sonstige Multimedia Objekte in die Datenbank eingebunden. Datenbanken klassischer Prägung und insbesondere auch relationale Datenbanken sind nur beschränkt in der Lage solche Objekte zu speichern, geschweige denn mit SQL-Kommandos abzufragen. Dennoch sind gerade diese Multimedia Objekte essentieller Bestandteil eines modernen Informationssystems.

Mit der Verbreitung des HTML Standards und der zunehmenden Nutzung des WWW (World Wide Web) durch Benutzer, die im Umgang mit Computern relativ unerfahren sind, sind die Anforderungen an solche "Multimedia Datenbanken" stetig gewachsen. In der Regel findet man im WWW keine relationalen Datenbanken, sondern meist eine Sammlung von HTML-Seiten, die durch Hyperlinks verbunden und mit Bildern, Filmen, Animationen und Tönen versehen sind. Im Raumfahrtbereich gibt es allerdings nur wenige Beispiele. Meist konzentrieren sich die wenigen verfügbaren "Datenbanken" auf spezielle Gebiete der Raumfahrt (Space Station, oder Produkte der jeweiligen Firma, die den WWW-Server betreibt).

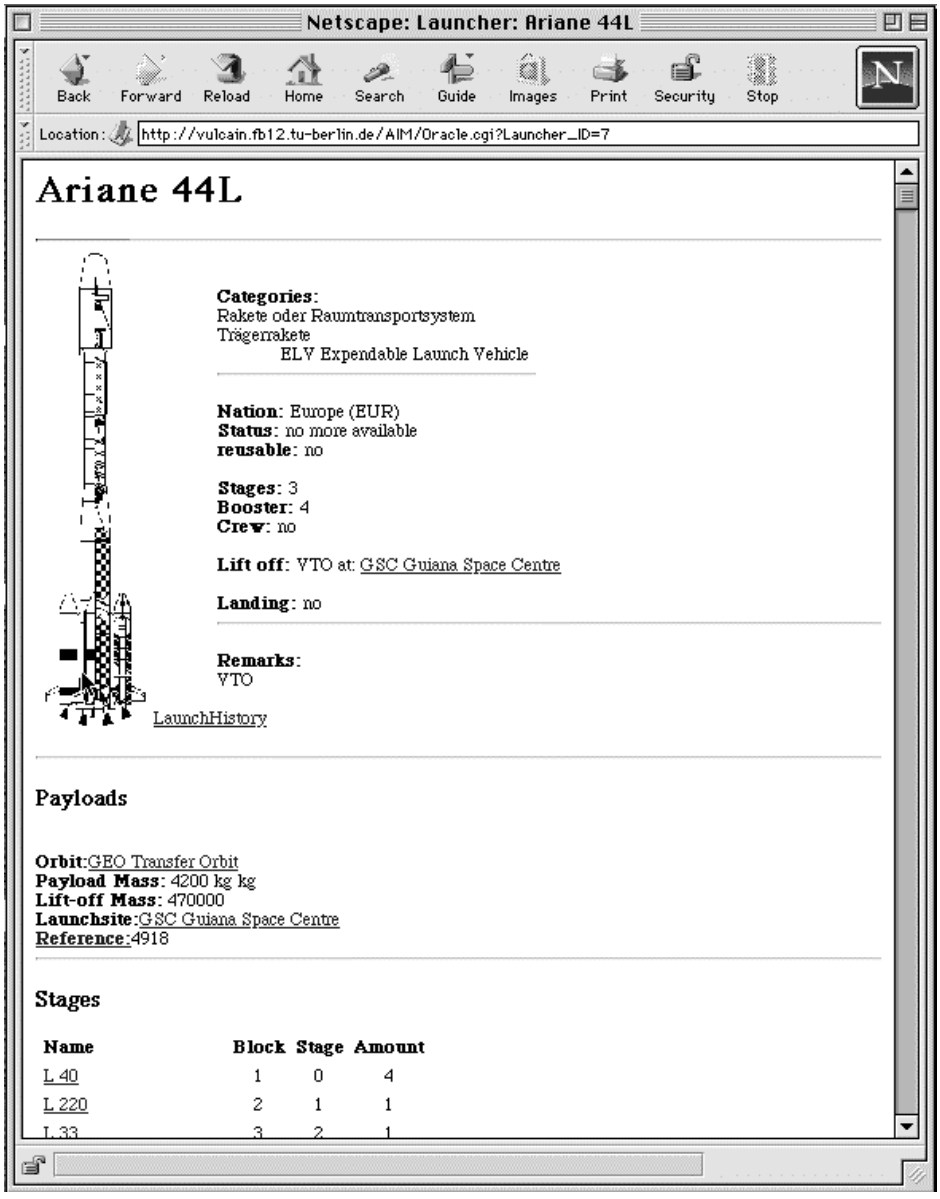


Abb. 7-6 Ergebnisdarstellung mit Multimedia (AIM Informationssystem WWW-Interface)

Abb. 7-6 zeigt ein Beispiel für die Einbindung von Bildern (es könnten genauso gut Animationen, Videos oder Töne sein) im Rahmen des AIM Informationssystems. Theoretisch ist es möglich jegliche Art von Multimedia Elementen, die sich in computerlesbaren Dateien speichern lassen in die AIM Datenbank zu integrieren. In der derzeitigen Fassung sind bereits einige Hundert Bilder enthalten. Die Abbildung zeigt das Ergebnis einer Datenbankanfrage über das WWW Interface des Informationssystems. Die sichtbaren Hyperlinks deuten an, dass alle Möglichkeiten des HTML Standards bei der Darstellung der Informationen genutzt werden. Dennoch handelt es sich nicht um eine reine Sammlung von HTML Seiten, sondern die Seiten werden online durch SQL-Kommandos an den Datenbank Server vom Interface erzeugt und dem WWW Client zur Verfügung gestellt.

Die natürliche Betrachtungsweise des Benutzers richtet sich jedoch nach der Realität oder aus Sicht des Datenbankentwurfs nach der Diskurswelt. Das heißt der Benutzer betrachtet die zu einem Objekt gehörenden Attribute, welche ja in den Spalten der einzelnen Tabellen abgelegt sind, als Einheit. Durch den Normalisierungsprozess, also die Aufspaltung der Daten auf mehrere Tabellen, bedingt durch die Überführung in die 3NF oder 4NF, ist die physikalische Nähe der Daten nicht mehr gegeben, so dass bei einer Darstellung der Informationen, wie in Abb. 7-3 gezeigt, eine große Anzahl von Tabellen mit den entsprechenden Daten angezeigt würde. In Abb. 7-2 würde eine sehr breite Tabelle mit sehr wenigen Zeilen als Resultat eines mitunter sehr komplizierten SQL-Kommandos das Ergebnis darstellen. Die Lösung wäre entweder eine selektive Auswahl bestimmter Attribute unter Verzicht auf eventuell wichtige Informationen oder, wie bei einigen Objekten des AIM Informationssystems geschehen, eine Zusammenstellung aller zu einem Objekt gehörenden Attribute durch sequentielle Anwendung mehrerer SQL-Kommandos in einer speziellen Client-Software. Die erhaltenen Daten müssen dann möglichst platzsparend in freier Anordnung dargestellt werden. Es obliegt dabei dem Programmierer festzulegen, welche Daten zu einem Objekt gehören und welche nicht. Die Abb. 7-7 zeigt ein solches Beispiel aus der AIM Datenbank, bei dem alle wichtigen Daten einer Literaturquelle gleichzeitig dargestellt werden. Die Daten sind auf mehrerer Tabellen verteilt und werden durch mehrere SQL-Kommandos gesammelt, aufbereitet und dargestellt. In modernen Entwicklungsumgebungen für Datenbanken können solche Ausgabemasken, auch "Forms" genannt, sehr schnell und bequem erzeugt werden.

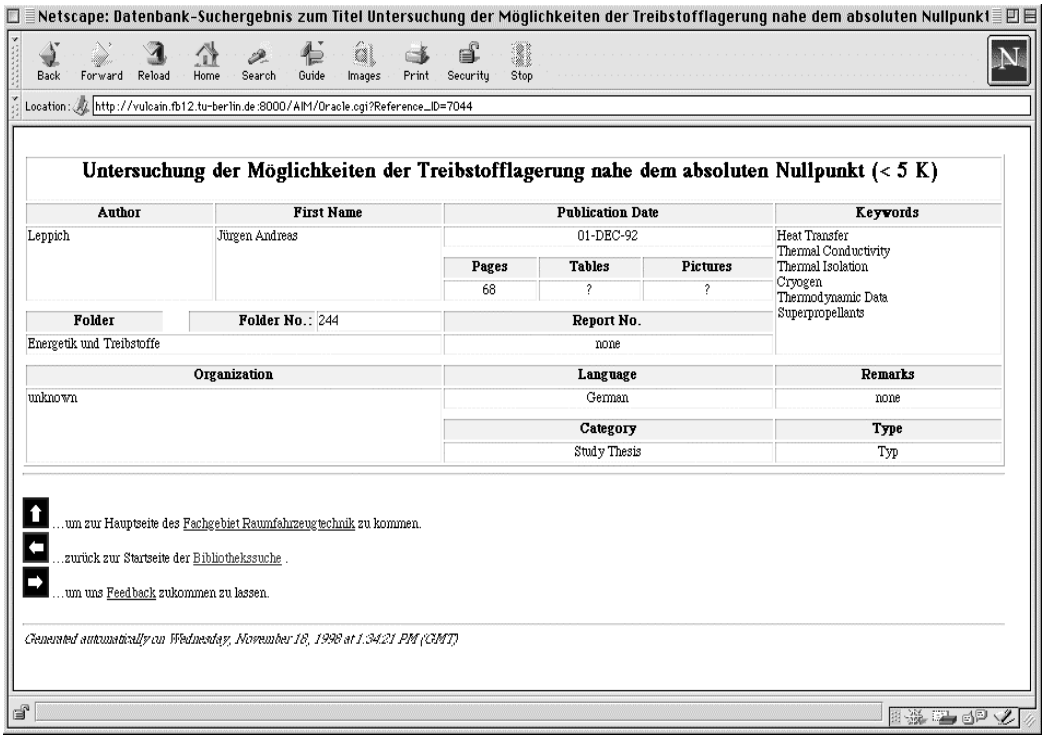


Abb. 7-7 Einzelobjektdarstellung: Alle Daten einer Literaturquelle

Zusätzlich zu den bereits erwähnten Vorteilen des WWW kommt noch die Möglichkeit durch Hyperlinks bzw. CGI Programmierung die unterschiedlichsten Zugriffsmethoden auf die Daten bereitzustellen. So ist die Benutzung des bereits in Kap 0 beschriebenen Schichtenmodells als Zugriffsstruktur mit mehreren Vorteilen verbunden. Der in der Raumfahrt erfahrene Benutzer erhält eine übersichtliche Einteilung der in der Datenbank vorhandenen Objekte. Über die verschiedenen Hyperlinks kann er sich innerhalb dieser Struktur relativ schnell zwischen den Schichten und Knoten bewegen (s.a. Abb. 7-8). Letztendlich landet er jedoch immer bei den Daten zu einem konkreten Objekt der Datenbank (wie in Abb. 7-6 für ein Objekt der Objektklasse Launcher gezeigt).

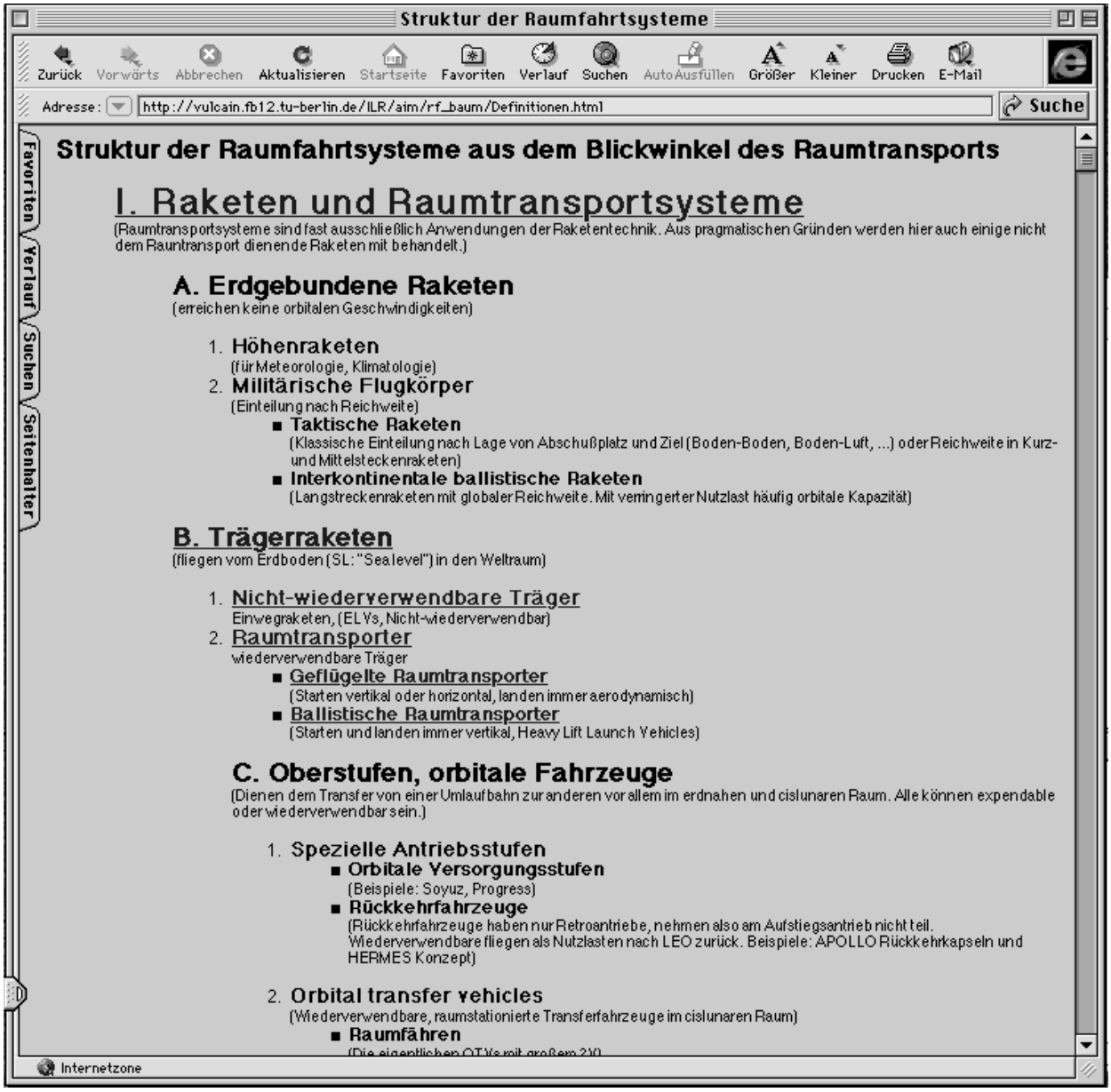


Abb. 7-8 AIM Schichtenmodell: Definitionsschicht

8 Zusammenfassung und Ausblick

In diesem abschließenden Kapitel soll noch einmal kurz zusammengefasst werden, was mit der vorliegenden Arbeit erreicht wurde. Der umfangreichere Ausblick soll zeigen, in welchen Bereichen noch Verbesserungen vorzunehmen sind und welche zukünftigen Entwicklungen in dem behandelten Gebiet zu erwarten sind.

8.1 Zusammenfassung

Bei der Durchführung eines Promotionsvorhabens geht es in erster Linie ums Prinzip und nicht unbedingt um die Erstellung funktionsfähiger Systeme. Die Machbarkeit bestimmter Vorgehensweisen soll prinzipiell gezeigt oder demonstriert werden. Insofern geht diese Arbeit über das Ziel hinaus, denn es ist gelungen ein frei zugängliches, funktionierendes Informationssystem zu schaffen, in dem nahezu alle in der Raumfahrt vorstellbaren Informationen auf übersichtliche Art und Weise gespeichert werden können. Dieses Informationssystem wird von einem Expertensystemteil ergänzt, der es erlaubt an Hand von Regelwissen das bestehende Faktenwissen zu ergänzen und zu überprüfen. Dieses einsatzfähige System hat mit ca. 400 Regeln den Prototypenstatus hinter sich gelassen und kann bei entsprechendem Ausbau und weiterer Geschwindigkeitssteigerung standardmäßig innerhalb des Informationssystems eingesetzt werden. Bei den verwendeten Expertenregeln hängt es sehr stark von dem bereits in der Datenbasis vorhandenen Bestand an Daten ab, ob differenzierte Aussagen gemacht werden können. Bei den in den Gesprächen mit den Experten gefundenen Regeln fielen die Voraussetzungen für bestimmte Regeln zum Teil sehr umfangreich aus. Ein Großteil dieser Regeln wurde noch nicht implementiert, da eine Anwendung, im Hinblick auf den noch nicht sehr umfangreichen Datenbestand, und damit wegen der fehlenden Voraussetzungen bei den meisten Objekten der Datenbank, als nicht sehr wahrscheinlich erschien. Zusätzlich würde die Abarbeitungsgeschwindigkeit der PROLOG Regeln durch die unnötig große Anzahl an Regeln nur verzögert werden.

Dennoch steht dem Wissenschaftler und den Studierenden ein Werkzeug zur Verfügung, welches für statistische Analysen und Modelle benötigte Daten schnell und einfach zur Verfügung stellt. Die besondere Anforderung in diesem Bereich nach möglichst vollständigen Datensätzen zu jedem Objekt wird durch den Expertensystemteil hervorragend erfüllt. Es wurde gezeigt, dass es auf einem zwar eng begrenzten, aber dennoch stark differenzierten Gebiet wie der Raumfahrt möglich ist, Faktenwissen und Regelwissen in einem System zu vereinen und innerhalb akzeptabler Antwortzeiten Informationen aus beiden Bereichen zur Verfügung zu stellen. Die Verfügbarkeit des Systems ist durch die Anwendung von Internettechniken weltweit gegeben. Durch die Verwendung von kommerziellen Standards (relationale Datenbank, SQL, WWW) wurde eine Insellösung vermieden und die Möglichkeit einer eventuellen Portierung offengehalten und der zu erwartende Aufwand verringert.

Im Bereich der Raumfahrt stellt das AIM System durch die Anwendung von KI-Methoden, die freie weltweite Verfügbarkeit und die Anwendung industrieller Datenbankstandards eine absolute Neuerung dar.

8.2 Ausblick

Auch die besten Systeme können noch besser gemacht werden und das AIM System bildet hier keine Ausnahme. Im folgenden sollen die verschiedenen Bereiche des Systems nach Verbesserungsmöglichkeiten untersucht werden. Außerdem werden Implementierungen vorgestellt, die angedacht waren, aber aus Zeitgründen nicht verwirklicht wurden.

Informationssystem

Das Hauptproblem beim Informationssystem sind die unvollständigen Datensätze der einzelnen Raumfahrtobjekte. Da der Expertensystemteil nicht alle Lücken schließen kann und um die Aktualität der

Daten (insbesondere Starthistorie und Neuentwicklungen) zu gewährleisten, ist eine Vereinfachung und wenn möglich Automatisierung der Dateneingabe vordringliches Ziel.

Datenbank

Beim Datenbankschema würde sich neben der bereits in Kap 5.2 angesprochenen Ausgliederung von Gruppen in Entity-Tabellen anbieten, die Attributbezeichnungen in einer oder mehreren (für jeden Objekttyp) eigenen Tabellen zu speichern. Dadurch wird eine einheitliche Bezeichnung erreicht und der Benutzer kann gezwungen werden nur Bezeichnungen aus diesen Tabellen zu verwenden. Außerdem wäre neben einer Erhöhung der Übersichtlichkeit eine leichtere Änderung einzelner Attributbezeichnungen möglich. Die Attribute könnten in verschiedenen Sprachen gespeichert werden, und es könnten Aliastabellen für alternative Attributbezeichnungen eingeführt werden, die ein Auffinden der Informationen unterstützen.

Dateneingabe

Neben der Dateneingabe von Hand, welche nur durch verstärkten Personaleinsatz gesteigert werden kann, bietet sich teilweise oder auch vollständige Automatisierung in den folgenden Bereichen an:

- Durchsuchen von eingescannten Texten oder anderen Quellen (WWW, etc.) nach verwendbaren Daten
- Automatische Übernahme von Simulationsergebnissen
- Dateneingabe über WWW ermöglichen
- Einbau von Simulationsprogrammen und/oder Iterationsprogrammen, welche Ergebnisse online produzieren (im Prinzip wie Regeln zu behandeln)

Abfragesoftware

Neben dem in Kap 7.1 geschilderten Idealfall einer Datenabfrage können natürlich auch naheliegendere Ziele verwirklicht werden:

- Fallbasierte Abfragetools (gleichartige Abfragen mit nur kleinen Unterschieden anhand von alten (leicht abgeänderten) Abfragen bearbeiten
- Anbieten aller Tabellen im WWW
- Eingabe von SQL-Kommandos über WWW
- Typische Abfragereihen (Protokolle einzelner Sitzungen) von bestimmten oder allen Benutzern analysieren und Lösungsstrategien für Abfragen erarbeiten

Darstellung der Ergebnisse

Bei der Darstellung der Ergebnisse von Datenbankabfragen können verschiedenen Möglichkeiten relativ leicht implementiert werden:

- Tabellarische Darstellung der Ergebnisse im WWW

  Um die Eigenschaften mehrerer Objekte für Vergleiche gleichzeitig darstellen zu können, müsste nur das bestehende CGI Programm etwas erweitert werden.
- Freie Definition der Ergebnisdarstellung durch den Benutzer

  Um dem Benutzer die Auswahl der anzuzeigenden Objekte und ihrer Eigenschaften zu gestatten, müsste schon etwas mehr Aufwand betrieben werden. Wahrscheinlich kommt man bei dieser Aufgabe um eine Applet Programmierung (Java, JavaSkript, Active X) nicht herum.

Expertensystem

Wissensakquisition

- Ausbau der Wissensbasis (Regeln für zusätzliche Objektklassen)
- Verbesserung der Warnungsfunktion bei Abschätzungen (mehrstufige tiefergehende Untersuchungen)
- Einbau einer Wissensakquisitionskomponente
- Implementierung automatischer Wissensakquisition (Anhand der Verbesserungen der Experten neue Regeln definieren). Zusätzlich kann bei Verbesserungen festhalten werden, was verbessert wurde, und bei mehreren gleichartigen Verbesserungen können daraus neue Regeln abgeleitet werden. Anhand der vorhandenen Daten können durch Erkennen von Übereinstimmungen zwischen Objektklassen neue Regeln abgeleitet werden.

Datenkonsistenz

- Relationen zur Überprüfung der Daten einsetzen
- Zusammenstellen von maximalen konsistenten Datensätzen mit Bedingung möglichst viele Daten aus der Faktenbasis
- Einbau von Wahrscheinlichkeiten anstelle von yes/no und damit Lösungsfindung auf Basis von unsicherem Wissen und Angabe von Wahrscheinlichkeiten für die verschiedenen Lösungen

Erklärungskomponente

- Grafische Darstellung des Lösungsbaums (Farbliche Kennzeichnung, Ein- und Ausblendung von Lösungspfaden)
- Grafische Darstellung von Formeln
- Detaillierte Erklärungen zu allen Regeln

Trotz der zahlreichen aufgeführten Verbesserungsmöglichkeiten wird das hier vorgestellte Informations- und Expertensystem hoffentlich viel Nutzen für die Raumfahrt bringen und in Zukunft noch ausgebaut und weiterentwickelt werden.

9 Literatur

[1] Baldacci, Frederic "Konzeption und Programmierung eines WWW-Interface für die Ermittlung von Emissionen von Raumfahrzeugen", Diplomarbeit am Institut für Luft- und Raumfahrt, TU Berlin, April 1997

[2] Bratko, Ivan: "Prolog - Programmierung für Künstliche Intelligenz"; Addison Wesley 1987

[3] Buchanan, B.G et.al. "Constructing an Expert System", in: "Building Expert Systems", Addison Wesley Publishing Co., London, 1983

[4] Cawsey, Alison "The Essence of Artificial Intelligence", Essence of Computing, Prentice Hall 1998

[5] Ceri, Stefano; Gottlob, Georg; Tanca, Letizia "Logic Programming and Databases", Surveys in Computer Science, Springer Verlag 1990

[6] Chen, P. P.-S. "The Entity-Relationship Model - Toward a Unified View of Data", in: ACM Transactions on Database Systems, Vol. 1, No.1, March 1976

[7] Codd, E. "Is Your DBMS Really Relational?", ComputerWorld, 14. Oktober 1985

[8] Codd, E. "Does Your DBMS Run By the Rules?", ComputerWorld, 21. Oktober 1985

[9] Ebert, Heinz "Scharfer Datenmix - Einführung in die Kunst des richtigen Datenbankentwurfs", in: c´t - Magazin für Computertechnik, Heft 9, September 1993

[10] Elmasri, R.; Navathe, S. "Fundamentals of Database Systems", Redwood City, The Benjamin/Cummings Publishing Co, 1994

[11] Ferstl, Otto K.; Sinz, Elmar J. "Glossar zum Begriffssystem des Semantischen Objektmodells (SOM)"; Bamberger Beiträge zur Wirtschaftsinformatik Nr. 11, Otto-Friedrich Universität Bamberg, Juni 1992

[12] Ferstl, Otto K.; Sinz, Elmar J. "Objektmodellierung betrieblicher Informationsmodelle im Semantischen Objektmodell (SOM)"; Bamberger Beiträge zur Wirtschaftsinformatik Nr. 1/1990, Otto-Friedrich Universität Bamberg, Juli 1990

[13] Ford, Nigel "So denken Maschinen – Einführung in die Künstliche Intelligenz am Beispiel von PROLOG", R. Oldenbourg Verlag, München und Wien 1988

[14] Gillenson, Mark L. "Datenbank-Konzepte, Datenbanken • Netzwerke • Expertensysteme"; 2. Auflage, Sybex Verlag GmbH Düsseldorf, 1991

[15] Ginsberg, Matt "Essentials of Artificial Intelligence", Morgan Kaufmann Publishers, San Francisco 1993

[16] Harmon, Paul; King, David "Expertensysteme in der Praxis", R. Oldenbourg Verlag, München und Wien 1986

[17] Heuer, Andreas "Objektorientierte Datenbanken", Addison Wesley,1992

[18] Hunter, Andrew "Relational Database Systems Handbook", Dept. of Computer Science, University of Sunderland, England, http://osiris.sunderland.ac.uk/ahu/

[19] Jackson, Peter "Introduction to Expert Systems", 2nd Edition, Addison Wesley 1990

[20] Johns, Nicky "MacDBI for Oracle Refrence Guide – Interface to Oracle for MacProlog32", Logic Programming Associates Ltd., March 1994

[21] Karbach, Werner; Linster, Marc "Wissensakquisition für Expertensysteme", Carl Hanser Verlag, München und Wien 1990

[22] Kurbel, Karl: "Entwicklung und Einsatz von Expertensystemen - Eine anwendungsorientierte Einführung in wissensbasierte Systeme"; Springer Verlag, 1992

[23] Laufer, René "Aufbau eines World Wide Web Servers für das Fachgebiet Raumfahrzeugtechnik", Studienarbeit am Institut für Luft- und Raumfahrt, TU Berlin, 1996

[24] Lausen, G.; Vossen, G. "Objektorientierte Datenbanken: Modelle und Sprachen", R.Oldenbourg-Verlag, 1996

[25] Leppich, Jürgen; Lo, Roger "German Database of Space Flight induced Atmospheric Pollution", International Symposium on the Impact of Emissions from Aircraft and Spacecraft on the Atmosphere

[26] Lo, Landvogt, Leppich, Sattler: "FESTIP WP2100", IRTS und IPK, 1997

[27] Lo, Roger; et. al.: "Raketentechnik", Skript zur Vorlesung, Institut für Luft- und Raumfahrt, TU-Berlin, 1995

[28] Lo, Roger; et. al.: "Raumfahrtantriebe"; Skript zur Vorlesung, Institut für Luft- und Raumfahrt, TU Berlin, 1995

[29] Lutz, Theo; Klimesch, Herbert "Die Datenbank im Informationssystem", R. Oldenbourg Verlag, München und Wien 1971

[30] N.N. "Database Administrators Guide for ORACLE Server for Macintosh-Version 6.0", ORACLE Corporation 1992

[31] N.N. "Advanced Performance Tuning for ORACLE Server for Macintosh-Version 1.0", ORACLE Corporation 1992

[32] N.N. "ORACLE Server for Macintosh-Getting Started-Version 6.0", ORACLE Corporation 1992

[33] N.N. "SQL Language Reference for ORACLE Server for Macintosh-Version 6.0", ORACLE Corporation 1992

[34] N.N. "SQL Language Reference for ORACLE Server for Macintosh", ORACLE Corporation, 1992

[35] N.N. http://webopedia.internet.com/

[36] N.N. http://www.co-soft.com/products/integra4/coddsrules.html

[37] N.N. "Codds Test", COMPUTERWOCHE Nr. 50 vom 13.Dezember 1991, Computerwoche Verlag GmbH, München

[38] Nork, Christian "Entwicklung von Datenmanipulationssoftware im Rahmen eines Raumfahrtinformationssystems (AIM)", Studienarbeit am Institut für Luft- und Raumfahrt, TU Berlin, 1995

[39] Paatsch, Alexander "Konzeptionierung und Programmierung von CGI-Software des WWW Server für das Fachgebiet Raumfahrzeugtechnik", Studienarbeit am Institut für Luft- und Raumfahrt, TU Berlin, Juli 1997

[40] Puppe, Frank "Einführung in Expertensysteme", Studienreihe Informatik, Springer Verlag 1988 u. 1991

[41] Rich, Elaine "KI-Einführung und Anwendungen", McGraw Hill Book Company, 1988

[42] Rich, Elaine "Artificial Intelligence", McGraw-Hill Book Company, New York, 1983

[43] Sauer, Hermann: "Relationale Datenbanken", Addison Wesley, 3. Ausgabe, 1994

[44] Savory, Stuart E. "Grundlagen von Expertensystemen", Nixdorf Computer AG, 2. Auflage, R.Oldenburg Verlag, München und Wien, 1990

[45] Schlageter, G.; Stucky, W. "Datenbanksysteme: Konzepte und Modelle", 2. Auflage, Teubner Verlag, Stuttgart 1983

[46] Schnupp, Peter; Leibrandt, Ute "Expertensysteme – Nicht nur für Informatiker",Springer Compass, Springer Verlag, 1986

[47] Sinz, Elmar J.: "Datenmodellierung im Strukturierten-Entity-Relationship-Modell (SERM)"; Bamberger Beiträge zur Wirtschaftsinformatik Nr. 10, Otto-Friedrich Universität Bamberg, Mai 1992

[48] Stede, Manfred; et. al. "Einführung in die Künstliche Intelligenz" "Band 1: Methodische Grundlagen", W.-D. Luther Verlag, Sprendlingen 1983

[49] Stede, Manfred; et. al. "Einführung in die Künstliche Intelligenz" "Band 2: Anwendungsgebiete", W.-D. Luther Verlag, Sprendlingen 1984

[50] Sterling, Leon; Shapiro, Ehud "The Art of Prolog: Advanced Programming Techniques", MIT Press, Cambridge, March 1987

[51] Sutton, George P.: "Rocket Propulsion Elements-An Introduction to the Engineering of Rockets"; 6. Auflage, John Wiley & Sons Inc., 1992

[52] Tanimoto, Steven L. "KI: Die Grundlagen", R. Oldenbourg Verlag, München und Wien 1990

[53] Tobehn, Carsten "Erstellen einer relationalen Quellendatenbasis im Rahmen eines Raumfahrtinformationssystems (AIM)", Studienarbeit am Institut für Luft- und Raumfahrt, TU Berlin, 1993

[54] Westwood, Dave "LPA Prolog 2.6 Technical Reference", Logic Programming Associates Ltd., 1994

[55] Zink, Fabian "Erstellung einer Regelbasis für ein Expertensystem der Raumfahrt", Diplomarbeit am Institut für Luft- und Raumfahrt, TU Berlin, 1998

10 Anhang

10.1 Tabellarische Übersicht aller AIM-Tabellen

Tabellen und Anzahl der Einträge

Tabelle	Einträge	Tabelle	Einträge	Tabelle	Einträge
Address	58	Launch_Reference	3357	Satellite_Picture	0
Attribute	269	LIQEngine	321	SpaceStation	13
Engine	722	Mission	4995	SpaceStation_Alias	19
Engine_Alias	145	Mission_Alias	1199	SpaceStation_Category	0
Engine_Category	3673	Mission_Category	4239	SpaceStation_Cost	3
Engine_Cost	1	Mission_Date	4725	SpaceStation_Date	32
Engine_Date	321	Mission_EVA	4	SpaceStation_Geometry	109
Engine_Emission	474	Mission_Failure	97	SpaceStation_Mass	45
Engine_Gasgenerator	22	Mission_Nation	3629	SpaceStation_Mission	12
Engine_Geometry	1116	Mission_Orbit	4077	SpaceStation_Module	43
Engine_Ignitor	13	Mission_Organization	92	SpaceStation_Nation	11
Engine_Injector	27	Mission_Person	68	SpaceStation_Organization	9
Engine_Mass	569	Mission_Satellite	61	SpaceStation_Performance	142
Engine_Organization	601	Module	37	SpaceStation_Picture	1
Engine_Performance	4043	Module_Alias	0	SpaceStation_Subsystem	29
Engine_Picture	118	Module_Cost	0	SpaceStation_TextFile	12
Engine_Pump	114	Module_Date	6	Stage	319
Engine_Reliability	6	Module_Geometry	51	Stage_Alias	23
Engine_TextFile	12	Module_Mass	35	Stage_Category	123
Engine_Turbine	59	Module_Nation	33	Stage_Cost	0
Engine_Valve	50	Module_Organization	0	Stage_Date	148
Fairing	165	Module_Performance	91	Stage_Engine	208
Fairing_Cost	0	Module_Picture	0	Stage_Geometry	401
Fairing_Geometry	5	Module_TextFile	7	Stage_Mass	1042
Fairing_Mass	169	Nation	244	Stage_Organization	118
Fairing_Organization	0	Nation_Alias	28	Stage_Performance	144
Fairing_Performance	14	Orbit	3902	Stage_Picture	7
Fairing_Picture	9	Orbit_GEO	26	Stage_Propellant	192
Folder	312	Organization	890	Stage_Reliability	0
Gasgenerator	29	Organization_Address	47	Stage_TextFile	0
Gasgenerator_Cost	0	Organization_Alias	1	Subject	36
Gasgenerator_Geometry	0	Organization_Nation	172	Subject_Keyword	897
Gasgenerator_Mass	11	Organization_Organization	6	Substance	392
Gasgenerator_Performance	56	Person	7137	Substance_Category	1242
Ignitor	20	Person_Address	7	Substance_Composition	127
Ignitor_Cost	0	Person_Alias	0	Substance_CubicalExpansion	23
Ignitor_Geometry	0	Person_Nation	170	Substance_Density	304
Ignitor_Mass	4	Person_Organization	101	Substance_Enthalpy	36
Ignitor_Performance	17	Person_Picture	0	Substance_HeatCapacity	62
Injector	28	Picture	369	Substance_HeatConductivity	54
Injector_Mass	18	Propellant	108	Substance_IsentropicExponent	49
Injector_Performance	0	Propellant_BurnRate	3	Substance_Property	685
Keyword	653	Propellant_Category	255	Substance_SurfaceTension	81
Launch	4078	Propellant_Emission	132928	Substance_TextFile	19
Launcher	312	Propellant_Substance	38	Substance_VaporPressure	315
Launcher_Alias	208	Pump	124	Substance_Viscosity	161
Launcher_Category	774	Pump_Mass	26	Subsystem	24
Launcher_Cost	21	Pump_Performance	277	Subsystem_Alias	1
Launcher_Date	18	Reference	7693	Subsystem_Cost	0
Launcher_Fairing	174	Reference_Folder	7314	Subsystem_Date	1
Launcher_Geometry	181	Reference_Keyword	27011	Subsystem_Geometry	4
Launcher_Mass	227	Reference_Organization	4044	Subsystem_Mass	42
Launcher_Nation	289	Reference_Person	11962	Subsystem_Nation	23
Launcher_Organization	128	Reference_Reference	181	Subsystem_Organization	1
Launcher_Payload	215	Reference_Rule	0	Subsystem_Performance	177
Launcher_Performance	13	Reference_Type	1811	Subsystem_Picture	0
Launcher_Picture	203	Rule	486	Subsystem_TextFile	0
Launcher_Reliability	0	Rule_Category	0	TextFile	60
Launcher_Stage	843	Rule_Condition	1077	Trajectory	79
Launcher_TextFile	54	Rule_Picture	0	Trajectory_Events	1148
Launcher_Trajectory	55	Rule_Reference	1	Trajectory_Vectors	19703
Launchsite	76	Rule_Result	486	Turbine	68
Launchsite_Alias	18	Rule_TextFile	0	Turbine_Mass	4
Launchsite_Category	0	Satellite	75	Turbine_Performance	206
Launchsite_Nation	92	Satellite_Alias	0	Valve	50
Launchsite_Organization	0	Satellite_Category	0	Valve_Cost	0
Launchsite_Picture	1	Satellite_Cost	11	Valve_Geometry	0
Launchsite_TextFile	0	Satellite_Date	7	Valve_Mass	19
Launch_Failure	276	Satellite_Geometry	67	Valve_Organization	19
Launch_Launcher	4972	Satellite_Mass	42	Valve_Performance	280
Launch_Orbit	120	Satellite_Organization	38		

Tabellen und Spalten

Tabelle	Spalte	Typ	Länge	Pr	Sc	Nu
Address	Address_ID	Number	22	5		N
	Nation_ID	Number	22	5		Y
	Zip_Code	Char	8			Y
	City	Char	30			Y
	Address_Remarks	Char	240			Y
	Street	Char	40			Y
Attribute	Attribute_ID	Number	22	5		Y
	Attribute_Name	Char	200			Y
	Attribute_Table	Char	48			Y
	Attribute_Class	Char	48			Y
	Attribute_Remarks	Char	240			Y
	Attribute_Column	Char	60			Y
	Attribute_GermanName	Char	200			Y
	Attribute_ShortName	Char	20			Y
	Attribute_Unit	Char	20			Y
Engine	Engine_ID	Number	22	5		N
	Propellant_ID	Number	22	5		Y
	Engine_Name	Char	30			Y
	Engine_ShortName	Char	15			Y
	Engine_Status	Char	25			Y
	Engine_Remarks	Char	240			Y
	Engine_ChamberCooling	Char	30			Y
	Engine_Restart	Char	25			Y
	Engine_Reuseability	Char	15			Y
	Engine_Suspension	Char	20			Y
Engine_Alias	Engine_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Engine_AliasName	Char	40			N
Engine_Category	Engine_ID	Number	22	5		N
	Engine_Category	Char	40			N
	Engine_CategoryLevel	Number	22	2		Y
Engine_Cost	Reference_ID	Number	22	5		N
	Engine_ID	Number	22	5		N
	Engine_CostType	Char	100			N
	Engine_CostRemarks	Char	240			Y
	Engine_CostValue	Number	22			Y
	Engine_CostUnit	Char	15			Y
	Engine_CostDefault	Char	1			Y
Engine_Date	Reference_ID	Number	22	5		N
	Engine_ID	Number	22	5		N
	Engine_DateType	Char	100			N
	Engine_DateRemarks	Char	240			Y
	Engine_DateDefault	Char	1			Y
	Engine_DateValue	Date	7			Y
Engine_Emission	Engine_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Substance_ID	Number	22	5		N
	Eng_Emismassfraction	Number	22			N
	Eng_Emissubstancestate	Char	1			N
	Eng_EmisDefault	Char	1			N
	Eng_Emismixtureratio	Number	22			N
Engine_Gasgenerator	Engine_ID	Number	22	5		N
	Engine_GgAmount	Number	22	2		N
	Gasgenerator_ID	Number	22	5		N
	Engine_GgType	Char	80			N
	Engine_GgRemarks	Char	250			Y
Engine_Geometry	Engine_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Engine_GeomType	Char	120			N
	Engine_GeomValue	Number	22			Y
	Engine_GeomRemarks	Char	240			Y
	Engine_GeomUnit	Char	3			Y
	Engine_GeomDefault	Char	1			Y
Engine_Ignitor	Engine_ID	Number	22	5		N
	Engine_IgnitorAmount	Number	22	2		N
	Ignitor_ID	Number	22	5		N
	Engine_IgnitorType	Char	80			N
	Engine_IgnitorRemarks	Char	250			Y
Engine_Injector	Engine_ID	Number	22	5		N
	Injector_ID	Number	22	5		N
Engine_Mass	Reference_ID	Number	22	5		N
	Engine_ID	Number	22	5		N
	Engine_MassType	Char	100			N
	Engine_MassValue	Number	22			Y
	Engine_MassRemarks	Char	240			Y
	Engine_MassDefault	Char	1			Y
Engine_Organization	Organization_ID	Number	22	5		N
	Engine_ID	Number	22	5		N
	Engine_OrgTask	Char	30			N
	Engine_OrgRemarks	Char	240			Y

Engine_Performance	Engine_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Engine_PerfType	Char	120			N
	Engine_PerfValue	Number	22			Y
	Engine_PerfDefault	Char	1			Y
	Engine_PerfRemarks	Char	240			Y
	Engine_PerfUnit	Char	12			Y
Engine_Picture	Picture_ID	Number	22	5		N
	Engine_ID	Number	22	5		N
Engine_Pump	Engine_ID	Number	22	5		N
	Pump_ID	Number	22	5		N
	Engine_PumpAmount	Number	22	2		N
	Engine_PumpType	Char	30			N
Engine_Reliability	Reference_ID	Number	22	5		N
	Engine_ID	Number	22	5		N
	Engine_Relrestarts	Number	22	4		N
	Engine_RelType	Char	25			N
	Engine_RelRemarks	Char	240			Y
	Engine_RelValue	Number	22			Y
	Engine_Reldate	Date	7			Y
	Engine_RelDefault	Char	1			Y
Engine_Textfile	Engine_ID	Number	22	5		N
	Textfile_ID	Number	22	5		Y
Engine_Turbine	Engine_ID	Number	22	5		N
	Turbine_ID	Number	22	5		N
	Engine_TurbineAmount	Number	22	2		N
	Engine_TurbineType	Char	30			N
Engine_Valve	Engine_ID	Number	22	5		N
	Engine_ValveAmount	Number	22	2		N
	Valve_ID	Number	22	5		N
	Engine_ValveType	Char	80			N
	Engine_ValveRemarks	Char	250			Y
Fairing	Fairing_ID	Number	22	5		N
	Fairing_Nopayloads	Number	22	2		Y
	Fairing_Category	Char	40			Y
	Fairing_Reuseability	Char	10			Y
	Fairing_ShortName	Char	10			Y
	Fairing_Name	Char	40			Y
	Fairing_Status	Char	20			Y
	Fairing_Remarks	Char	240			Y
Fairing_Cost	Fairing_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Fairing_CostType	Char	25			N
	Fairing_CostRemarks	Char	240			Y
	Fairing_CostUnit	Char	15			Y
	Fairing_CostDefault	Char	1			Y
	Fairing_CostValue	Number	22			Y
Fairing_Geometry	Fairing_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Fairing_GeomType	Char	25			N
	Fairing_GeomValue	Number	22			Y
	Fairing_GeomDefault	Char	1			Y
	Fairing_GeomRemarks	Char	240			Y
Fairing_Mass	Fairing_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Fairing_MassType	Char	25			N
	Fairing_MassValue	Number	22			Y
	Fairing_MassRemarks	Char	240			Y
	Fairing_MassDefault	Char	1			Y
Fairing_Organization	Fairing_ID	Number	22	5		N
	Organization_ID	Number	22	5		N
	Fairing_OrgTask	Char	30			N
	Fairing_OrgRemarks	Char	240			Y
Fairing_Performance	Fairing_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Fairing_PerfType	Char	25			N
	Fairing_PerfUnit	Char	10			N
	Fairing_PerfValue	Number	22			Y
	Fairing_PerfRemarks	Char	240			Y
	Fairing_PerfDefault	Char	1			Y
Fairing_Picture	Fairing_ID	Number	22	5		N
	Picture_ID	Number	22	5		N
Folder	Folder_ID	Number	22	5		N
	Folder_Name	Char	80			Y
	Folder_Remarks	Char	240			Y
Gasgenerator	Gasgenerator_ID	Number	22	5		N
	Gasgenerator_Name	Char	40			Y
	Gasgenerator_Remarks	Char	255			Y
Gasgenerator_Cost	Gasgenerator_CostType	Char	200			N
	Gasgenerator_CostDefault	Char	1			N
	Gasgenerator_CostValue	Number	22			N
	Gasgenerator_ID	Number	22	5		Y
	Reference_ID	Number	22	5		Y
	Gasgenerator_CostUnit	Char	20			Y
	Gasgenerator_CostRemarks	Char	250			Y

Gasgenerator_Geometry	Gasgenerator_GeomType	Char	200			N
	Gasgenerator_GeomDefault	Char	1			N
	Gasgenerator_GeomValue	Number	22			N
	Gasgenerator_ID	Number	22	5		Y
	Reference_ID	Number	22	5		Y
	Gasgenerator_GeomUnit	Char	15			Y
	Gasgenerator_GeomRemarks	Char	250			Y
Gasgenerator_Mass	Gasgenerator_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Gasgenerator_MassType	Char	200			N
	Gasgenerator_MassDefault	Char	1			N
	Gasgenerator_MassValue	Number	22			N
	Gasgenerator_MassRemarks	Char	250			Y
Gasgenerator_Performance	Gasgenerator_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Gasgenerator_PerfType	Char	120			N
	Gasgenerator_PerfValue	Number	22			Y
	Gasgenerator_PerfRemarks	Char	240			Y
	Gasgenerator_PerfDefault	Char	1			Y
	Gasgenerator_PerfUnit	Char	12			Y
Ignitor	Ignitor_ID	Number	22	5		N
	Ignitor_Name	Char	40			Y
	Ignitor_Remarks	Char	255			Y
Ignitor_Cost	Ignitor_CostType	Char	200			N
	Ignitor_CostDefault	Char	1			N
	Ignitor_CostValue	Number	22			N
	Ignitor_ID	Number	22	5		Y
	Reference_ID	Number	22	5		Y
	Ignitor_CostUnit	Char	20			Y
	Ignitor_CostRemarks	Char	250			Y
Ignitor_Geometry	Ignitor_GeomType	Char	200			N
	Ignitor_GeomDefault	Char	1			N
	Ignitor_GeomValue	Number	22			N
	Ignitor_ID	Number	22	5		Y
	Reference_ID	Number	22	5		Y
	Ignitor_GeomUnit	Char	15			Y
	Ignitor_GeomRemarks	Char	250			Y
Ignitor_Mass	Ignitor_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Ignitor_MassType	Char	200			N
	Ignitor_MassValue	Number	22			N
	Ignitor_MassDefault	Char	1			N
	Ignitor_MassRemarks	Char	250			Y
Ignitor_Performance	Ignitor_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Ignitor_PerfType	Char	120			N
	Ignitor_PerfValue	Number	22			Y
	Ignitor_PerfDefault	Char	1			Y
	Ignitor_PerfRemarks	Char	240			Y
	Ignitor_PerfUnit	Char	12			Y
Injector	Injector_ID	Number	22	5		N
	Injector_Name	Char	50			Y
	Injector_Remarks	Char	255			Y
	Injector_Type	Char	25			Y
Injector_Mass	Injector_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Injector_MassType	Char	25			N
	Injector_MassValue	Number	22			N
	Injector_MassDefault	Char	1			N
	Injector_MassRemarks	Char	240			Y
Injector_Performance	Injector_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Injector_PerfType	Char	120			N
	Injector_PerfValue	Number	22			N
	Injector_PerfUnit	Char	20			Y
	Injector_PerfDefault	Char	1			Y
	Injector_PerfRemarks	Char	240			Y
Keyword	Keyword_ID	Number	22	5		N
	Keyword	Char	100			N
	Old_ID	Number	22	5		Y
	Stichwort	Char	50			Y
Launch	Launch_ID	Number	22	5		N
	Launch_Nopayloads	Number	22	2		Y
	Launch_Stagesinorbit	Number	22	2		Y
	Launch_Masspayloads	Number	22	12	4	Y
	Launch_Date	Date	7			Y
	Launch_Time	Date	7			Y
	Launch_Remarks	Char	240			Y
	Launch_Result	Char	15			Y

Launcher	Launcher_ID	Number	22	5		N
	No_Stages	Number	22	2		Y
	No_Boosters	Number	22	2		Y
	Launcher_Name	Char	40			Y
	Launcher_Family	Char	50			Y
	Launcher_Crewed	Char	8			Y
	Landing_Mode	Char	2			Y
	Launcher_Remarks	Char	240			Y
	Launcher_Status	Char	20			Y
	Liftoff_Mode	Char	3			Y
	Launcher_Reuseability	Char	10			Y
	Launcher_Version	Char	20			Y
Launcher_Alias	Launcher_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Launcher_AliasName	Char	80			N
Launcher_Category	Launcher_ID	Number	22	5		N
	Launcher_Category	Char	40			N
	Launcher_CategoryLevel	Number	22	2		Y
Launcher_Cost	Launcher_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Launcher_CostType	Char	100			N
	Launcher_CostRemarks	Char	240			Y
	Launcher_CostValue	Number	22			Y
	Launcher_CostUnit	Char	15			Y
	Launcher_CostDefault	Char	1			Y
Launcher_Date	Launcher_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Launcher_DateType	Char	100			N
	Launcher_DateRemarks	Char	240			Y
	Launcher_DateDefault	Char	1			Y
	Launcher_DateValue	Date	7			Y
Launcher_Fairing	Launcher_ID	Number	22	5		N
	Fairing_ID	Number	22	5		N
Launcher_Geometry	Launcher_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Launcher_GeomType	Char	25			N
	Launcher_GeomValue	Number	22			Y
	Launcher_GeomUnit	Char	3			Y
	Launcher_GeomRemarks	Char	240			Y
	Launcher_GeomDefault	Char	1			Y
Launcher_Mass	Launcher_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Launcher_MassType	Char	100			N
	Launcher_MassValue	Number	22			Y
	Launcher_MassRemarks	Char	240			Y
	Launcher_MassDefault	Char	1			Y
Launcher_Nation	Launcher_ID	Number	22	5		N
	Nation_ID	Number	22	5		N
	Nation_Activity	Char	240			Y
	Launcher_NationDefault	Char	1			Y
Launcher_Organization	Organization_ID	Number	22	5		N
	Launcher_ID	Number	22	5		N
	Launcher_OrgTask	Char	30			N
	Launcher_OrgRemarks	Char	240			Y
Launcher_Payload	Launcher_ID	Number	22	5		N
	Launchsite_ID	Number	22	5		N
	Orbit_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Payload_Remarks	Char	240			Y
	Payload_Default	Char	1			Y
	Mliftoff	Number	22			Y
	Payload	Number	22			Y
Launcher_Performance	Launcher_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Launcher_PerfValue	Number	22			N
	Launcher_PerfType	Char	25			N
	Launcher_PerfRemarks	Char	240			Y
	Launcher_PerfDefault	Char	1			Y
	Launcher_PerfUnit	Char	15			Y
Launcher_Picture	Launcher_ID	Number	22	5		N
	Picture_ID	Number	22	5		N
Launcher_Reliability	Launcher_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Launcher_Rellaunches	Number	22	4		N
	Launcher_RelType	Char	25			N
	Launcher_RelValue	Number	22			Y
	Launcher_RelRemarks	Char	240			Y
	Launcher_Reldate	Date	7			Y
	Launcher_RelDefault	Char	1			Y
Launcher_Stage	Launcher_ID	Number	22	5		N
	Stage_ID	Number	22	5		N
	Launcher_Stageno	Number	22	2		Y
	Launcher_StageAmount	Number	22	2		Y
	Launcher_Stageblockno	Number	22	2		Y
	Launcher_StageRemarks	Char	240			Y

Launcher_Textfile	Launcher_ID	Number	22	5		N
	Textfile_ID	Number	22	5		N
Launcher_Trajectory	Trajectory_ID	Number	22	5		N
	Launcher_ID	Number	22	5		N
	Orbit_ID	Number	22	5		N
	Launchsite_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Launcher_TrajectoryDefault	Char	1			N
	Launch_Azimuth	Number	22			N
Launchsite	Launchsite_ID	Number	22	5		N
	No_Launchpads	Number	22	2		Y
	No_Landingranges	Number	22	2		Y
	Launchsite_Utildaysperyear	Number	22	3		Y
	Launchsite_Area	Number	22	12	4	Y
	Launchsite_Cost	Number	22	12	4	Y
	Launchsite_Maxvelocitygain	Number	22	12	4	Y
	Longitude	Number	22			Y
	Azimuth_Min	Number	22			Y
	Latitude	Number	22			Y
	Launchsite_Name	Char	50			Y
	Launchsite_Status	Char	20			Y
	Launchsite_Operationstart	Date	7			Y
	Launchsite_Remarks	Char	240			Y
	Altitude	Number	22			Y
	Launchsite_ShortName	Char	8			Y
	Longitude_Dir	Char	1			Y
	Latitude_Dir	Char	1			Y
	Azimuth_Max	Number	22			Y
Launchsite_Alias	Launchsite_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Launchsite_AliasName	Char	40			N
Launchsite_Category	Launchsite_ID	Number	22	5		N
	Launchsite_Category	Char	25			N
	Launchsite_CategoryLevel	Number	22	2		Y
Launchsite_Nation	Launchsite_ID	Number	22			N
	Nation_ID	Number	22			N
	Lsite_Nationdescription	Char	15			N
Launchsite_Organization	Launchsite_ID	Number	22	5		N
	Organization_ID	Number	22	5		N
	Lsite_OrgTask	Char	30			N
	Lsite_OrgRemarks	Char	240			Y
Launchsite_Picture	Launchsite_ID	Number	22	5		N
	Picture_ID	Number	22	5		N
Launchsite_Textfile	Launchsite_ID	Number	22	5		N
	Textfile_ID	Number	22	5		N
Launch_Failure	Launch_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Launcher_ID	Number	22	5		Y
	Launch_Failurestage	Number	22	2		Y
	Launch_Failurepart	Char	25			Y
	Launch_Failurephase	Char	20			Y
	Launch_FailureRemarks	Char	240			Y
Launch_Launcher	Launch_ID	Number	22	5		N
	Launcher_ID	Number	22	5		N
	Launchsite_ID	Number	22	5		Y
	Reference_ID	Number	22	5		Y
	Fairing_ID	Number	22	5		Y
	Launch_LauncherDefault	Char	1			Y
Launch_Orbit	Orbit_ID	Number	22	5		N
	Launch_ID	Number	22	5		N
	Trajectory_ID	Number	22	5		Y
	Launch_OrbitType	Char	25			Y
	Launch_Azimuth	Number	22			Y
Launch_Reference	Launch_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
Liqengine	Engine_ID	Number	22	5		N
	Engine_Nopumps	Number	22	1		Y
	Engine_Noturbines	Number	22	1		Y
	Liqengine_Nocombchambers	Number	22	2		Y
	Liqengine_Noverniers	Number	22	2		Y
	Engine_Nogasgen	Number	22	1		Y
	Engine_Cycle	Char	30			Y
	Liqengine_IspLevel	Char	8			Y
	Liqengine_ThrustLevel	Char	8			Y

Mission	Mission_ID	Number	22	5		N
	Launch_ID	Number	22	5		Y
	Mission_Crewed	Number	22	2		Y
	Mission_Lifeduration	Number	22	4		Y
	Mission_Noradnumber	Number	22	5		Y
	Mission_Intdesignation	Char	15			Y
	Mission_Liftoffmass	Number	22			Y
	Mission_Result	Char	15			Y
	Mission_Function	Char	40			Y
	Mission_Name	Char	50			Y
	Mission_Remarks	Char	240			Y
	Mission_Status	Char	15			Y
	Mission_OrbitType	Char	5			Y
	Mission_Estlifeduration	Char	15			Y
Mission_Alias	Mission_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Mission_AliasName	Char	80			N
Mission_Category	Mission_ID	Number	22	5		N
	Mission_Category	Char	50			N
	Mission_CategoryLevel	Number	22	2		Y
Mission_Date	Mission_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Mission_DateType	Char	25			N
	Mission_DateValue	Date	7			N
	Mission_DateDefault	Char	1			N
Mission_Eva	Mission_ID	Number	22	5		N
	Person_ID	Number	22	5		N
	Eva_Date	Date	7			N
	Orbit_ID	Number	22	5		Y
	Evaduration	Number	22	4		Y
	Evalocation	Char	10			Y
	EvaRemarks	Char	240			Y
Mission_Failure	Mission_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Mission_FailureRemarks	Char	240			Y
	Mission_Failuredescription	Char	50			Y
	Mission_Failureresult	Char	50			Y
Mission_Nation	Mission_ID	Number	22	5		N
	Nation_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Mission_NationDefault	Char	1			N
Mission_Orbit	Mission_ID	Number	22	5		N
	Orbit_ID	Number	22	5		N
	Orbit_Date	Date	7			N
	Mission_OrbitRemarks	Char	240			Y
	Mission_OrbitType	Char	50			Y
Mission_Organization	Mission_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Organization_ID	Number	22	5		N
	Mission_OrgDefault	Char	1			N
	Mission_OrgTask	Char	40			Y
	Mission_OrgRemarks	Char	240			Y
Mission_Person	Mission_ID	Number	22	5		N
	Person_ID	Number	22	5		N
	Mission_Persfunction	Char	20			Y
	Mission_PersRemarks	Char	240			Y
Mission_Satellite	Mission_ID	Number	22	5		N
	Satellite_ID	Number	22	5		N
Module	Module_ID	Number	22	5		N
	Module_Name	Char	40			N
	Module_Dockingadapters	Number	22	2		Y
	Module_Status	Char	20			Y
	Module_Remarks	Char	254			Y
Module_Alias	Module_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Module_Alias	Char	50			N
Module_Cost	Module_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Module_CostType	Char	25			N
	Module_CostDefault	Char	1			N
	Module_CostUnit	Char	1			N
	Module_CostValue	Number	22			N
	Module_CostRemarks	Char	254			Y
Module_Date	Module_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Module_DateType	Char	25			N
	Module_DateValue	Date	7			N
	Module_DateDefault	Char	1			N
	Module_DateRemarks	Char	254			Y
Module_Geometry	Module_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Module_GeomType	Char	25			N
	Module_GeomDefault	Char	1			N
	Module_GeomValue	Number	22			N
	Module_GeomRemarks	Char	254			Y

Module_Mass	Module_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Module_MassType	Char	25			N
	Module_MassDefault	Char	1			N
	Module_MassValue	Number	22			N
	Module_MassRemarks	Char	254			Y
Module_Nation	Module_ID	Number	22	5		N
	Nation_ID	Number	22	5		N
	Module_NationTask	Char	25			N
Module_Organization	Module_ID	Number	22	5		N
	Organization_ID	Number	22	5		N
	Module_OrgTask	Char	25			N
	Module_OrgRemarks	Char	254			Y
Module_Performance	Module_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Module_PerfType	Char	25			N
	Module_PerfDefault	Char	1			N
	Module_PerfUnit	Char	15			N
	Module_PerfValue	Number	22			N
	Module_PerfRemarks	Char	254			Y
Module_Picture	Module_ID	Number	22	5		N
	Picture_ID	Number	22	5		N
Module_Textfile	Module_ID	Number	22	5		N
	Textfile_ID	Number	22	5		N
Nation	Nation_ID	Number	22	5		N
	Nation_Isonumber	Number	22	3		Y
	Nation_Remarks	Char	240			Y
	Nation_Name	Char	80			Y
	Nation_Capital	Char	30			Y
	Nation_A2Name	Char	2			Y
	Nation_ShortName	Char	3			Y
	Nation_PostName	Char	20			Y
	Nation_Phonenumber	Char	4			Y
Nation_Alias	Nation_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Nation_AliasName	Char	80			N
Orbit	Orbit_ID	Number	22	5		N
	Argofperigee	Number	22			Y
	Lenofascnode	Number	22			Y
	Perigee	Number	22			Y
	Orbit_Name	Char	50			Y
	Orbit_Remarks	Char	240			Y
	Orbit_Type	Char	20			Y
	Orbit_Centralbody	Char	20			Y
	Semiaxis	Number	22			Y
	Orbit_ShortName	Char	5			Y
	Apogee	Number	22			Y
	Inclination	Number	22			Y
	Eccentricity	Number	22			Y
	Period	Number	22			Y
Orbit_Geo	Orbit_ID	Number	22	5		N
	Geo_Position	Number	22			Y
	Long_Dir	Char	1			Y
Organization	Organization_ID	Number	22	5		N
	Organization_Name	Char	100			Y
	Organization_ShortName	Char	12			Y
	Organization_Category	Char	50			Y
	Organization_Remarks	Char	240			Y
	Organization_Activities	Char	240			Y
Organization_Address	Address_ID	Number	22	5		N
	Organization_ID	Number	22	5		N
	Dept_Name	Char	100			N
	Dept_Phone	Char	12			Y
	Dept_Telex	Char	30			Y
	Dept_Activities	Char	240			Y
	Dept_Email	Char	30			Y
	Dept_Fax	Char	12			Y
	Dept_ShortName	Char	12			Y
	Dept_Remarks	Char	240			Y
Organization_Alias	Organization_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Organization_AliasName	Char	80			N
Organization_Nation	Organization_ID	Number	22	5		N
	Nation_ID	Number	22	5		N
	Org_Comments	Char	240			Y
Organization_Organization	Memberorg_ID	Number	22	5		N
	Organization_ID	Number	22	5		N
	Org_Notes	Char	240			Y

Person	Person_ID	Number	22	5		N
	Pers_Remarks	Char	240			Y
	Pers_Fax	Char	15			Y
	Pers_Phone	Char	15			Y
	Date_Death	Date	7			Y
	Pers_FirstName	Char	50			Y
	Date_Birth	Date	7			Y
	Pers_Name	Char	30			Y
	Pers_Email	Char	40			Y
	Pers_Title	Char	20			Y
Person_Address	Address_ID	Number	22	5		N
	Person_ID	Number	22	5		N
	Address_Remarks	Char	240			Y
Person_Alias	Person_ID	Number	22	5		N
	Person_AliasName	Char	50			N
	Person_AliasfirstName	Char	30			Y
Person_Nation	Nation_ID	Number	22	5		N
	Person_ID	Number	22	5		N
Person_Organization	Person_ID	Number	22	5		N
	Organization_ID	Number	22	5		N
	Dept_ShortName	Char	15			N
	Position	Char	50			Y
	Job_Email	Char	40			Y
	Dept_Name	Char	255			Y
	Job_Remarks	Char	240			Y
	Job_Fax	Char	15			Y
	Job_Phone	Char	15			Y
Person_Picture	Picture_ID	Number	22	5		N
	Person_ID	Number	22	5		N
Picture	Picture_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Picture_Name	Char	240			N
	Picture_FileName	Char	32			N
	Picture_Width	Number	22	4		Y
	Picture_Heigth	Number	22	4		Y
	Picture_Format	Char	20			Y
	Picture_Application	Char	20			Y
	Picture_Remarks	Char	240			Y
Propellant	Propellant_ID	Number	22	5		N
	Propellant_Name	Char	40			N
	Propellant_Storability	Char	20			Y
	Propellant_Remarks	Char	255			Y
	Propellant_State	Char	20			Y
Propellant_Burnrate	Propellant_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Propellant_BurnrateValue	Number	22			N
	Propellant_BurnrateDefault	Char	1			N
	Propellant_Burnratepressure	Number	22			Y
	Propellant_BurnrateRemarks	Char	240			Y
Propellant_Category	Propellant_ID	Number	22	5		N
	Propellant_Category	Char	40			N
Propellant_Emission	Reference_ID	Number	22	5		N
	Propellant_ID	Number	22	5		N
	Substance_ID	Number	22	5		N
	Prop_Emismixtureratio	Number	22			N
	Prop_Emismassfraction	Number	22			N
	Prop_Emismolefraction	Number	22			N
	Prop_Emissubstancestate	Char	1			N
	Prop_EmisDefault	Char	1			N
	Prop_Emisexpansionratio	Number	22			N
	Prop_Emischamberpressure	Number	22			N
	Prop_Emisarearatio	Number	22			Y
	Prop_Emisexittemperature	Number	22			Y
Propellant_Substance	Substance_ID	Number	22	5		N
	Propellant_ID	Number	22	5		N
	Substance_Purpose	Char	20			Y
	Substance_State	Char	15			Y
	Prop_SubstRemarks	Char	240			Y
Pump	Pump_ID	Number	22	5		N
	Pump_Nostages	Number	22	2		Y
	Pump_Name	Char	50			Y
	Pump_Type	Char	25			Y
	Pump_Remarks	Char	255			Y
Pump_Mass	Pump_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Pump_MassType	Char	25			N
	Pump_MassValue	Number	22			N
	Pump_MassDefault	Char	1			N
	Pump_MassRemarks	Char	240			Y
Pump_Performance	Pump_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Pump_PerfType	Char	120			N
	Pump_PerfValue	Number	22			Y
	Pump_PerfRemarks	Char	240			Y
	Pump_PerfDefault	Char	1			Y
	Pump_PerfUnit	Char	20			Y
Reference	Reference_ID	Number	22	5		N

	Ref_Title	Char	240			N
	Ref_Nopages	Number	22	4		Y
	Ref_Notables	Number	22	4		Y
	Ref_Nopictures	Number	22	4		Y
	Ref_Category	Char	15			Y
	Ref_Publicationdate	Date	7			Y
	Ref_Remarks	Char	240			Y
	Ref_Language	Char	15			Y
	Ref_Reportno	Char	25			Y
Reference_Folder	Reference_ID	Number	22	5		N
	Folder_ID	Number	22	5		N
Reference_Keyword	Reference_ID	Number	22	5		N
	Keyword_ID	Number	22	5		N
Reference_Organization	Reference_ID	Number	22	5		N
	Organization_ID	Number	22	5		N
	Org_Function	Char	30			Y
Reference_Person	Reference_ID	Number	22	5		N
	Person_ID	Number	22	5		N
	Reference_Personnumber	Number	22	2		Y
	Person_Task	Char	30			Y
Reference_Reference	Volume_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
Reference_Rule	Reference_ID	Number	22	5		N
	Rule_ID	Number	22	5		N
	Reference_Ruleorder	Number	22	2		N
Reference_Type	Reference_ID	Number	22	5		N
	Type	Char	20			N
Rule	Rule_ID	Number	22	5		N
	Rule_Name	Char	200			N
	Rule_Confidence	Number	22	2		Y
	Rule_Remarks	Char	250			Y
Rule_Category	Rule_ID	Number	22	5		N
	Rule_Category	Char	20			N
Rule_Condition	Rule_ID	Number	22	5		N
	Rule_Conditionclass	Char	100			N
	Rule_Conditionattribut	Char	250			N
Rule_Picture	Rule_ID	Number	22	5		N
	Picture_ID	Number	22	5		N
Rule_Reference	Rule_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
Rule_Result	Rule_ID	Number	22	5		N
	Rule_Resultclass	Char	100			N
	Rule_Resultattribut	Char	250			N
Rule_Textfile	Rule_ID	Number	22	5		N
	Textfile_ID	Number	22	5		N
Satellite	Satellite_ID	Number	22	5		N
	Satellite_Category	Char	30			Y
	Satellite_Shape	Char	25			Y
	Satellite_Remarks	Char	240			Y
	Satellite_Name	Char	80			Y
	Satellite_Status	Char	35			Y
Satellite_Alias	Satellite_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Satellite_AliasName	Char	80			N
Satellite_Category	Satellite_ID	Number	22	5		N
	Satellite_Category	Char	25			N
	Satellite_CategoryLevel	Number	22	2		Y
Satellite_Cost	Satellite_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Satellite_CostType	Char	25			N
	Satellite_CostUnit	Char	15			Y
	Satellite_CostDefault	Char	1			Y
	Satellite_CostValue	Number	22			Y
	Satellite_CostRemarks	Char	240			Y
Satellite_Date	Satellite_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Satellite_DateType	Char	30			N
	Satellite_DateDefault	Char	1			Y
	Satellite_DateValue	Date	7			Y
	Satellite_DateRemarks	Char	240			Y
Satellite_Geometry	Satellite_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Satellite_GeomType	Char	25			N
	Satellite_GeomRemarks	Char	240			Y
	Satellite_GeomDefault	Char	1			Y
	Satellite_GeomValue	Number	22			Y
Satellite_Mass	Satellite_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Satellite_MassType	Char	25			N
	Satellite_MassValue	Number	22			Y
	Satellite_MassRemarks	Char	240			Y
	Satellite_MassDefault	Char	1			Y

Satellite_Organization	Satellite_ID	Number	22	5		N
	Organization_ID	Number	22	5		N
	Satellite_OrgTask	Char	30			N
	Satellite_OrgRemarks	Char	240			Y
Satellite_Picture	Picture_ID	Number	22	5		N
	Satellite_ID	Number	22	5		N
Spacestation	Spacestation_ID	Number	22	5		N
	Spacestation_Name	Char	40			N
	Spacestation_Numberofmodules	Number	22	2		Y
	Spacestation_Dockingadapters	Number	22	2		Y
	Spacestation_Secondcrew	Number	22	2		Y
	Spacestation_Firstcrew	Number	22	2		Y
	Spacestation_Status	Char	20			Y
	Spacestation_Remarks	Char	254			Y
Spacestation_Alias	Spacestation_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Spacestation_Alias	Char	100			N
Spacestation_Category	Spacestation_ID	Number	22	5		N
	Spacestation_Category	Char	25			N
	Spacestation_CategoryLevel	Number	22	2		Y
Spacestation_Cost	Spacestation_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Spacestation_CostType	Char	200			N
	Spacestation_CostDefault	Char	1			N
	Spacestation_CostUnit	Char	5			N
	Spacestation_CostValue	Number	22			N
	Spacestation_CostRemarks	Char	254			Y
Spacestation_Date	Spacestation_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Spacestation_DateType	Char	200			N
	Spacestation_DateValue	Date	7			N
	Spacestation_DateDefault	Char	1			N
	Spacestation_DateRemarks	Char	240			Y
Spacestation_Geometry	Spacestation_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Spacestation_GeomType	Char	200			N
	Spacestation_GeomDefault	Char	1			N
	Spacestation_GeomValue	Number	22			N
	Spacestation_GeomRemarks	Char	254			Y
Spacestation_Mass	Spacestation_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Spacestation_MassType	Char	200			N
	Spacestation_MassValue	Number	22			N
	Spacestation_MassDefault	Char	1			N
	Spacestation_MassRemarks	Char	254			Y
Spacestation_Mission	Spacestation_ID	Number	22	5		N
	Mission_ID	Number	22	5		N
Spacestation_Module	Spacestation_ID	Number	22	5		N
	Module_ID	Number	22	5		N
Spacestation_Nation	Spacestation_ID	Number	22	5		N
	Nation_ID	Number	22	5		N
	Spacestation_NationTask	Char	25			N
	Spacestation_NationRemarks	Char	254			Y
Spacestation_Organization	Spacestation_ID	Number	22	5		N
	Organization_ID	Number	22	5		N
	Spacestation_OrgTask	Char	25			N
	Spacestation_OrgRemarks	Char	254			Y
Spacestation_Performance	Spacestation_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Spacestation_PerfType	Char	200			N
	Spacestation_PerfValue	Number	22			N
	Spacestation_PerfUnit	Char	15			N
	Spacestation_PerfDefault	Char	1			N
	Spacestation_PerfRemarks	Char	254			Y
Spacestation_Picture	Spacestation_ID	Number	22	5		N
	Picture_ID	Number	22	5		N
Spacestation_Subsystem	Spacestation_ID	Number	22	5		N
	Subsystem_ID	Number	22	5		N
	Spacestation_SubsystemAmount	Number	22	3		Y
Spacestation_Textfile	Spacestation_ID	Number	22	5		N
	Textfile_ID	Number	22	5		N
Stage	Stage_ID	Number	22	5		N
	Stage_Name	Char	50			Y
	Stage_Reuseability	Char	15			Y
	Stage_Separation	Char	30			Y
	Stage_Status	Char	20			Y
	Stage_Controlsystem	Char	15			Y
	Stage_Remarks	Char	240			Y
Stage_Alias	Stage_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Stage_AliasName	Char	80			N
Stage_Category	Stage_ID	Number	22	5		N
	Stage_Category	Char	25			N
	Stage_CategoryLevel	Number	22	2		Y

Stage_Cost	Reference_ID	Number	22	5		N
	Stage_ID	Number	22	5		N
	Stage_CostType	Char	100			N
	Stage_CostValue	Number	22			Y
	Stage_CostDefault	Char	1			Y
	Stage_CostRemarks	Char	240			Y
	Stage_CostUnit	Char	15			Y
Stage_Date	Reference_ID	Number	22	5		N
	Stage_ID	Number	22	5		N
	Stage_DateType	Char	240			N
	Stage_DateRemarks	Char	240			Y
	Stage_DateDefault	Char	1			Y
	Stage_DateValue	Date	7			Y
Stage_Engine	Stage_ID	Number	22	5		N
	Engine_ID	Number	22	5		N
	Stage_EngineAmount	Number	22	2		Y
	Stage_EngineRemarks	Char	240			Y
Stage_Geometry	Stage_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Stage_GeomType	Char	150			N
	Stage_GeomDefault	Char	1			N
	Stage_GeomUnit	Char	3			N
	Stage_GeomValue	Number	22			Y
	Stage_GeomRemarks	Char	240			Y
Stage_Mass	Reference_ID	Number	22	5		N
	Stage_ID	Number	22	5		N
	Stage_MassType	Char	100			N
	Substance_ID	Number	22	5		Y
	Stage_MassRemarks	Char	240			Y
	Stage_MassValue	Number	22			Y
	Stage_MassDefault	Char	1			Y
Stage_Organization	Organization_ID	Number	22	5		N
	Stage_ID	Number	22	5		N
	Stage_OrgTask	Char	30			N
	Stage_OrgRemarks	Char	240			Y
Stage_Performance	Reference_ID	Number	22	5		N
	Stage_ID	Number	22	5		N
	Stage_PerfType	Char	240			N
	Stage_PerfRemarks	Char	240			Y
	Stage_PerfUnit	Char	12			Y
	Stage_PerfValue	Number	22			Y
	Stage_PerfDefault	Char	1			Y
Stage_Picture	Picture_ID	Number	22	5		N
	Stage_ID	Number	22	5		N
Stage_Propellant	Stage_ID	Number	22	5		N
	Propellant_ID	Number	22	5		N
Stage_Reliability	Reference_ID	Number	22	5		N
	Stage_ID	Number	22	5		N
	Stage_Rellaunches	Number	22	4		N
	Stage_RelType	Char	25			N
	Stage_RelValue	Number	22			Y
	Stage_Reldate	Date	7			Y
	Stage_RelRemarks	Char	240			Y
	Stage_RelDefault	Char	1			Y
Stage_Textfile	Stage_ID	Number	22	5		N
	Textfile_ID	Number	22	5		N
Subject	Subject_ID	Number	22	3		N
	Subject	Char	50			N
	Sachgebiet	Char	50			Y
Subject_Keyword	Subject_ID	Number	22	3		N
	Keyword_ID	Number	22	5		N
Substance	Substance_ID	Number	22	5		N
	Substance_ShortName	Char	50			Y
	Substance_CommonName	Char	50			Y
	Substance_Odour	Char	50			Y
	Substance_Color	Char	50			Y
	Substance_Chemformula	Char	30			Y
	Substance_Name	Char	100			Y
	Substance_Remarks	Char	240			Y
Substance_Category	Substance_ID	Number	22	5		N
	Substance_Category	Char	25			N
Substance_Composition	Substance_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Substance_CompositionCompFrac	Number	22			N
	Substance_CompositionDefault	Char	1			N
	Substance_CompositionComponent	Char	50			Y
	Substance_CompositionRemarks	Char	240			Y
Substance_CubicalExpansion	Substance_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Substance_CubicalExpValue	Number	22			N
	Substance_CubicalExpDefault	Char	1			N
	Substance_CubicalExpTemp	Number	22			Y
	Substance_CubicalExpRemarks	Char	240			Y

Substance_Density	Substance_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Substance_DensityValue	Number	22			N
	Substance_DensityDefault	Char	1			N
	Substance_DensityTemp	Number	22			Y
	Substance_DensityRemarks	Char	240			Y
Substance_Enthalpy	Substance_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Substance_EnthalpyValue	Number	22			N
	Substance_EnthalpyDefault	Char	1			N
	Substance_EnthalpyTemp	Number	22			Y
	Substance_EnthalpyRemarks	Char	240			Y
Substance_HeatCapacity	Substance_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Substance_HeatCapacityValue	Number	22			N
	Substance_HeatCapacityDefault	Char	1			N
	Substance_HeatCapacityTemp	Number	22			Y
	Substance_HeatCapacityRemarks	Char	240			Y
Substance_HeatConductivity	Substance_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Substance_HeatCondValue	Number	22			N
	Substance_HeatCondDefault	Char	1			N
	Substance_HeatCondTemp	Number	22			Y
	Substance_HeatCondRemarks	Char	240			Y
Substance_IsentropicExponent	Substance_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Substance_IsentropicExpValue	Number	22			N
	Substance_IsentropicExpDefault	Char	1			N
	Substance_IsentropicExpTemp	Number	22			Y
	Substance_IsentropicExpRemarks	Char	240			Y
Substance_Property	Substance_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Substance_PropertyType	Char	200			Y
	Substance_PropertyValue	Number	22			Y
	Substance_PropertyUnit	Char	10			Y
	Substance_PropertyDefault	Char	1			Y
	Substance_PropertyRemarks	Char	240			Y
Substance_SurfaceTension	Substance_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Substance_SurfaceTensValue	Number	22			N
	Substance_SurfaceTensDefault	Char	1			N
	Substance_SurfaceTenstemp	Number	22			Y
	Substance_SurfaceTensRemarks	Char	240			Y
Substance_Textfile	Substance_ID	Number	22	5		N
	Textfile_ID	Number	22	5		N
Substance_VaporPressure	Substance_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Substance_VaporPressValue	Number	22			N
	Substance_VaporPressDefault	Char	1			N
	Substance_VaporPresstemp	Number	22			Y
	Substance_VaporPressRemarks	Char	240			Y
Substance_Viscosity	Substance_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Substance_ViscosityValue	Number	22			N
	Substance_ViscosityDefault	Char	1			N
	Substance_Viscositytemp	Number	22			Y
	Substance_ViscosityRemarks	Char	240			Y
Subsystem	Subsystem_ID	Number	22	5		N
	Subsystem_Name	Char	200			N
	Subsystem_Category	Char	20			Y
	Subsystem_Status	Char	20			Y
	Subsystem_Remarks	Char	254			Y
Subsystem_Alias	Subsystem_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Subsystem_Alias	Char	50			N
Subsystem_Cost	Subsystem_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Subsystem_CostType	Char	25			N
	Subsystem_CostUnit	Char	1			N
	Subsystem_CostValue	Number	22			N
	Subsystem_CostDefault	Char	1			N
	Subsystem_CostRemarks	Char	254			Y
Subsystem_Date	Subsystem_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Subsystem_DateType	Char	25			N
	Subsystem_DateDefault	Char	1			N
	Subsystem_DateValue	Date	7			N
	Subsystem_DateRemarks	Char	254			Y
Subsystem_Geometry	Subsystem_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Subsystem_GeomType	Char	60			N
	Subsystem_GeomDefault	Char	1			N
	Subsystem_GeomValue	Number	22			N
	Subsystem_GeomRemarks	Char	254			Y

Subsystem_Mass	Subsystem_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Subsystem_MassType	Char	25			N
	Subsystem_MassValue	Number	22			N
	Subsystem_MassDefault	Char	1			N
	Subsystem_MassRemarks	Char	254			Y
Subsystem_Nation	Subsystem_ID	Number	22	5		N
	Nation_ID	Number	22	5		N
	Subsystem_NationTask	Char	25			Y
	Subsystem_NationRemarks	Char	254			Y
Subsystem_Organization	Subsystem_ID	Number	22	5		N
	Organization_ID	Number	22	5		N
	Subsystem_OrgTask	Char	25			Y
	Subsystem_OrgRemarks	Char	254			Y
Subsystem_Performance	Subsystem_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Subsystem_PerfType	Char	50			N
	Subsystem_PerfValue	Number	22			N
	Subsystem_PerfDefault	Char	1			N
	Subsystem_PerfUnit	Char	12			Y
	Subsystem_PerfRemarks	Char	254			Y
Subsystem_Picture	Subsystem_ID	Number	22	5		N
	Picture_ID	Number	22	5		N
Subsystem_Textfile	Subsystem_ID	Number	22	5		N
	Textfile_ID	Number	22	5		N
Textfile	Textfile_ID	Number	22	5		N
	Reference_ID	Number	22	5		Y
	Textfile_Name	Char	32			Y
	Textfile_Remarks	Char	240			Y
	Textfile_Type	Char	15			Y
	Textfile_Size	Number	22			Y
Trajectory	Trajectory_ID	Number	22	5		N
	Trajectory_Name	Char	100			Y
	Trajectory_Remarks	Char	240			Y
Trajectory_Events	Trajectory_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Trajectory_Time	Number	22			N
	Event	Char	240			N
	Launcher_Stageblockno	Number	22	2		Y
	Event_Remarks	Char	240			Y
Trajectory_Vectors	Trajectory_ID	Number	22	5		N
	Trajectory_Time	Number	22			N
	Trajectory_Altitude	Number	22			Y
	Trajectory_Longitude	Number	22			Y
	Trajectory_Latitude	Number	22			Y
	Trajectory_Velocity	Number	22			Y
	Trajectory_Groundrange	Number	22			Y
	Trajectory_Localpitchangle	Number	22			Y
	Trajectory_Locationx	Number	22			Y
	Trajectory_Locationy	Number	22			Y
	Trajectory_Locationz	Number	22			Y
	Trajectory_Velocityx	Number	22			Y
	Trajectory_Velocityy	Number	22			Y
	Trajectory_Velocityz	Number	22			Y
	Trajectory_Attitudex	Number	22			Y
	Trajectory_Attitudey	Number	22			Y
	Trajectory_Attitudez	Number	22			Y
Turbine	Turbine_ID	Number	22	5		N
	Turbine_Nostages	Number	22	2		Y
	Turbine_Name	Char	50			Y
	Turbine_Remarks	Char	255			Y
	Turbine_Type	Char	30			Y
Turbine_Mass	Turbine_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Turbine_MassType	Char	25			N
	Turbine_MassValue	Number	22			N
	Turbine_MassDefault	Char	1			N
	Turbine_MassRemarks	Char	240			Y
Turbine_Performance	Turbine_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Turbine_PerfType	Char	120			N
	Turbine_PerfValue	Number	22			Y
	Turbine_PerfUnit	Char	12			Y
	Turbine_PerfRemarks	Char	240			Y
	Turbine_PerfDefault	Char	1			Y
Valve	Valve_ID	Number	22	5		N
	Valve_Name	Char	80			N
	Valve_Remarks	Char	255			Y
Valve_Cost	Valve_CostType	Char	200			N
	Valve_CostDefault	Char	1			N
	Valve_CostValue	Number	22			N
	Valve_ID	Number	22	5		Y
	Reference_ID	Number	22	5		Y
	Valve_CostUnit	Char	20			Y
	Valve_CostRemarks	Char	250			Y

Valve_Geometry	Valve_GeomType	Char	200			N
	Valve_GeomDefault	Char	1			N
	Valve_GeomValue	Number	22			N
	Valve_ID	Number	22	5		Y
	Reference_ID	Number	22	5		Y
	Valve_GeomUnit	Char	15			Y
	Valve_GeomRemarks	Char	250			Y
Valve_Mass	Valve_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Valve_MassType	Char	200			N
	Valve_MassValue	Number	22			N
	Valve_MassDefault	Char	1			N
	Valve_MassRemarks	Char	250			Y
Valve_Organization	Valve_ID	Number	22	5		N
	Organization_ID	Number	22	5		N
	Valve_OrgTask	Char	100			N
	Valve_OrgRemarks	Char	250			Y
Valve_Performance	Valve_ID	Number	22	5		N
	Reference_ID	Number	22	5		N
	Valve_PerfType	Char	120			N
	Valve_PerfValue	Number	22			Y
	Valve_PerfUnit	Char	12			Y
	Valve_PerfDefault	Char	1			Y
	Valve_PerfRemarks	Char	240			Y

10.2 Objekte und Attribute

Launcher

Name (englsich)	Einheit	Abkürzng	Name (deutsch)
Abolishment Cost	MJ	CA	Entsorgungskosten eines Trägers
Average Launch Rate	N/a	nstart	Startrate
Baufaktor	-	Mnet_M8	Baufaktor
Burn-Out Mass	kg	Mc	BrennSchlussmasse
Coarse Structure Mass	kg	Ms	Strukturmasse bei grober Betrachtung
Conceptual Design End	-	Phase Ae	Ende der Konzeptentwurfsphase
Conceptual Design Start	-	Phase A	Beginn der Konzeptentwurfsphase
Crewed	-	crewed	Bemenscht
Customer	-	custom	Kunde
DeltavGEO	m/s	dvGEO	Geschwindigkeitsbedarf GEO
DeltavLEO	m/s	dvLEO	Geschwindigkeitsbedarf LEO
Developer	-	dev	Entwicklungsorganisation
Development Cost	MJ	CD	Entwicklungskosten
Development End	-	Phase De	Entwicklungsende
Development manager	-	chefdev	Leiter der Entwicklung
Development Start	-	Phase D	Entwicklungsbeginn
Direct Operation Cost	MJ	DOC	Direkte Missionskosten
Dry Mass	kg	Mdry	Trockenmasse
Family	-	fam	Familie
Final Design Review	-	FDR	Abnahme der Entwicklung
Final Documentation/Report	-	FR	Enddokumentationsbericht
First Flight Attempt	-	FFA	Erster Flugversuch
Groß Factor	-	M0_M1	Grundverhältnis
Growth Factor	-	M0_Ms	Growth Factor
Guidance and Control Mass	kg	M2	Lenk- und Regelausrüstung
Indirect Operation Cost	MJ	IOC	Indirekte Missionskosten
Landing Mode	-	modeland	Landemodus
Last Flight	-	LF	Letzter Flug
Launch Cost	MJ	CL	Startkosten eines Trägers
Launch service agency	-	organlaunch	Startorganisation
Launch Site and Range Cost	MJ	LSC	Bodeninfrastrukturkosten
Launchsite	-		Startort
Life Cycle Cost	MJ	LCC	Lebensdauerkosten
Lift-Off Mass	kg	M0	Startmasse
Lift-off Mode	-	modestart	Startmodus
Main designer	-	fdes	Führende Designorganisation
Main developer	-	Fdev	Führende Entwicklungsorganisation
Maintenance and Refurbishment Cost	MJ	MRC	Wartungs- und Ersetzungskosten
Manufacturer	-	Manu	Fertigungsorganisation
Manufacturing Cost	MJ	CF	Fertigungskosten eines Trägers
Maximum Diameter	m	da,max	Max. Außendurchmesser
Name	-	name	Name
Net Mass	kg	Mnet	Netto Masse
Number Of Boosters	-	nboost	Anzahl der Booster
Number of Launcher	-	Nträger	Anzahl gebauter Trägersysteme
Number Of Stages	-	nstage	Anzahl der Stufen
Operation Cost	MJ	COPS	Missionskosten
Overall Length	m	ltot	Gesamtlänge
Overall Mass Ratio	-	R	Massenverhältnis
Payload Mass	kg	M1	Nutzlast
Payload Ratio	-	M1_M0	Nutzlastverhältnis
Prelimenary Design End	-	Phase Be	Ende der Vorentwurfsphase
Prelimenary Design Start	-	Phase B	Beginn der Vorentwurfsphase
Production End	-	Phase Ee	Produktionsende
Production Start	-	Phase E	Produktionsbeginn
Program/Project Definition End	-	Phase Ce	Ende der Entwurfsphase
Program/Project Definition Start	-	Phase C	Beginn der Entwurfsphase
Propellant Ratio	-	M8_M0	Treibstoffverhältnis
Propellant Residuals Mass	kg	M6	Reste von Treibstoffen und Gasen
Propulsion System Mass	kg	M4	Antriebssystem und Zubehör
Recovery System Mass	kg	M5	Bergungssystem
Reuseability	-	reuse	Wiederverwendbarkeit
Status	-	status	Status
Structure Mass	kg	M3	Zelle und Ausrüstung
Structure Ratio	-	Ms_M0	Strukturverhälnis
Succesful Flight Demonstration	-	SFD	Erste erfolgreiche Flugdemonstration
Usable Extra Propellant Mass	kg	M7	Nutzbare Treibstoffreserven
Usable Propellant Mass	kg	M8	Nutzbare Treibstoffmasse
Vehicle Amortization Cost	MJ	VAC	Fahrzeug-Amortisationskosten
Vehicle operator	-	Operat	Betreiber des Raumfahrzeugs
Version	-	vers	Version

Stage

Name (englisch)	Einheit	Abkürzung	Name (deutsch)
Actuator System Mass	kg	mact	Masse der Stellglieder für das TW-Schwenken
Baufaktor	-	mnet_m8	Baufaktor
Burn-Out Mass	kg	mc	BrennSchlussmasse
Conceptual Design End	-	Phase Ae	Ende der Konzeptentwurfsphase
Conceptual Design Start	-	Phase A	Beginn der Konzeptentwurfsphase
Developer	-	dev	Entwicklungsorganisation
Development End	-	Phase De	Entwicklungsende
Development Start	-	Phase D	Entwicklungsbeginn
Dry Mass	kg	mdry	Trockenmasse
Equipment Bay Diameter	m	dVEB	Durchmesser der Ausrüstungsbucht
Equipment Bay Length	m	lVEB	Länge der Ausrüstungsbucht
Equipment Bay Mass	kg	mVEB	Masse der Ausrüstungsbucht
Final Burn Time Test	-	FIT	Letzter Versuch mit max Brenndauer
Final Design Review	-	FBTT	Abnahme der Entwicklung
Final Documentation/Report	-	FR	Enddokumentationsbericht
First Flight Attempt	-	FFA	Erster Flugversuch
First Ignition Test	-	FDR	Erster Zündversuch
Fuel Tank Diameter	m	dfueltank	Durchmesser des Treibstofftanks
Fuel Tank Length	m	lfueltank	Länge des Treibstofftanks
Fuel Tank Pressure	Pa	pfueltank	Treibstofftankdruck
Fuel Tank Volume	m^2	Vfueltank	Treibstofftankvolumen
Groß Mass	kg	mgroß	Masse der Stufen auf der Rampe
Growth Factor	-	m0_ms	Growth Factor
Guidance and Control Mass	kg	m2	Lenk- u. Regelausrüstung
Interstage Adapter Mass	kg	mISA	Masse des Stufentrennungsadapters
Last Flight	-	LF	Letzter Flug
Lift-Off Mass	kg	m0	Startmasse
Main Developer	-	fdev	Führende Entwicklungsorganisation
Main Manufacturer	-	fmanu	Hauptfertigungsorganisation
Manufacturer	-	manu	Fertigungsorganisation
Net Mass	kg	mnet	Netto Masse
Overall Mass Ratio	-	R	Massenverhältnis
Oxidizer Tank Diameter	m	doxtank	Durchmesser des Oxidatortanks
Oxidizer Tank Length	m	loxtank	Länge des Oxidatortanks
Oxidizer Tank Pressure	Pa	poxtank	Oxidatortankdruck
Oxidizer Tank Volume	m^2	Voxtank	Oxidatortankvolumen
PL Interface Diameter	m	dPLinterf	Durchmesser der Nutzlastverbindung
Prelimenary Design End	-	Phase Be	Ende der Vorentwurfsphase
Prelimenary Design Start	-	Phase B	Beginn der Vorentwurfsphase
Pressurization Gas Mass	kg	mgas	Masse des Bedrückungsgasas
Pressurization Gas Tank Pressure	Pa	pPG	Druck des Bedrückungsgases
Prime Contractor	-	Pcontr	Hauptvertragspartner
Production End	-	Phase Ee	Produktionsende
Production Start	-	Phase E	Produktionsbeginn
Program/Project Definition End	-	Phase Ce	Ende der Entwurfsphase
Program/Project Definition Start	-	Phase C	Beginn der Entwurfsphase
Propellant Fuel Mass	kg	mfuel	Brennstoffmasse
Propellant Other Mass	kg	msonst	Additivmasse
Propellant Oxidizer Mass	kg	mox	Oxidatormasse
Propellant Ratio	-	m8_m0	Treibstoffverhältnis
Propellant Residuals Mass	kg	m6	Reste von Treibstoffen u. Gasen
Propellant Total Mass	kg	m8tot	Gesamttreibstoffmasse
Propulsion System Mass	kg	m4	Antriebssystem und Zubehör
Recovery System Mass	kg	m5	Bergungsausrüstung
Stage Accelaration Max	m/s^2	amax	maximale Stufenbeschleunigung
Stage Diameter Max	m	dstage	maximale Stufendurchmesser
Stage Length	m	lstage	Länge der Stufe
Stage Thrust	N	Fstage	Schub der Stufe
Structure Mass	kg	m3	Zelle und Ausrüstung
Structure Ratio	-	ms_m0	Strukturverhälnis
Subcontractor	-	Scontr	Sub-Vertragspartner
Succesful Flight Demonstration	-	SFD	Erste erfolgreiche Flugdemonstration
Tank Diameter	m	dtank	Durchmesser des Tanks
Usable Extra Propellant Mass	kg	m7	Nutzbare Treibstoffreserven
Usable Propellant Mass	kg	m8	ErwarteterTreibstoffverbrauch
VEB Interface Diameter	m	dVEBinterf	Durchmesser der Verbindung zur Ausrüstungsbucht

Engine

Name (englisch)	Einheit	Abkürzung	Name (deutsch)
Abbrandgeschwindigkeit	m/s	rpunkt	Abbrandgeschwindigkeit
Abbrandoberfläche	m^2	Ap	Abbrandoberfläche
Adiabatenexponent	-	Kappa	Adiabatenexponent
Aktionszeit	s	Za	Aktionszeit
Burn Time	s	tc	Brenndauer
Burn Time Maximum	s	tc,max	Maximale Brenndauer
Burn Time Per Mission	s	tc,per miss	Missionsabhängige Brennzeit
Chamber Combustion Pressure	pa	pc	Brennkammerdruck
Chamber Combustion Temperature	K	Tc	Brennkammertemperatur
Chamber Diameter	m	dchamb	Brennkammerdurchmesser
Chamber Fuel Inlet Pressure	pa	pfuelinl	Einlaufdruck des Treibstoffs
Chamber Fuel Inlet Pressure Maximum	pa	pfuelinl,max	Maximaler Einlaufdruck des Treibstoffs
Chamber Fuel Inlet Pressure Minimum	pa	pfuelinl,min	Minimaler Einlaufdruck des Treibstoffs
Chamber Fuel Inlet Temperature	K	Tfuelinl	Einspritztemperatur des Treibstoffs
Chamber Inlet Pressure	pa	pchamlinl	Einspritzdruck der Brennkammer
Chamber Inlet Pressure Maximum	pa	pchaminl,max	Max. Einspritzdruck der Brennkammer
Chamber Inlet Pressure Minimum	pa	pchaminl,min	Min. Einspritzdruck der Brennkammer
Chamber Length	m	lchamb	Brennkammerlänge
Chamber Mass	kg	mcham	Brennkammermasse
Chamber Mixture Ratio	-	O_Fcham	Mischungsverhältnis in der Brennkammer
Chamber Oxidizer Inlet Pressure	pa	poxinl	Einspritzdruck des Oxidators
Chamber Oxidizer Inlet Pressure Maximum	pa	poxinl,max	Maximaler Einspritzdruck des Oxidators
Chamber Oxidizer Inlet Pressure Minimum	pa	poxinl,min	Minimaler Einspritzdruck des Oxidators
Chamber Oxidizer Inlet Temperature	K	Toxinl	Einspritztemperatur des Oxidators
Chamber Wall Temperature Maximum	K	Tmax,chamwall	Maximale Temperatur der Brennkammerwand
Characteristic Length	m	lstern	Charakteristische Triebwerkslänge
Characteristic Velocity	m/s	cstern	Charakteristische Geschwindigkeit
Conceptual Design End	-	Phase Ae	Ende der Konzeptentwurfsphase
Conceptual Design Start	-	Phase A	Beginn der Konzeptentwurfsphase
Controller Mass	kg	mcontr	Masse desSteuerungssystems
Convergent Chamber Mass	kg	mcham,conv	Konvergente Brennkammermasse
Coolant Mass Flow Rate	kg/s	mpunktcool	Kühlmittelmassenstrom
Cylindrical Chamber Mass	kg	mcham,zyl	Zylindrische Brennkammermasse
Deflection Maximum	¡	alphadefl	Abweichungswinkel
Developer	-	dev	Entwicklungsorganisation
Development Start	-	Phase D	Entwicklungsbeginn
Dichte der Abgase am Düsenende	kg/m^2	rohe	Dichte der Abgase am Düsenende
Divergent Chamber Mass	kg	mcham,div	Divergente Brennkammermasse
Dry Mass	kg	mdry	Trockenmasse
Efficiency	-	nü	Wirkungsgrad
Engine Diameter	m	hengine	Triebwerksdurchmesser
Engine Height	m	dengine	Triebwerkshöhe
Engine Length	m	lengine	Treibwerkslänge
Engine Regulation Range (Lower Limit)	%	Engreg,min	Steuerungsbereich (unteres Limit)
Engine Regulation Range (Mixture Ratio)	%	Engreg,mid	Steuerungsbereich (Mittelwert)
Engine Regulation Range (Upper Limit)	%	Engreg,max	Steuerungsbereich (oberes Limit)
Engine Width	m	bengine	Triebwerksbreite
Exhaust Velocity	m/s	ce	Ausströmgeschwindigkeit
Feed System Mass	kg	mfeedsy	Masse des Fördersystems (=TW-DÜ-BK)
Final Documentation/Report	-	FR	Enddokumentationsbericht
First Flight Test	-	FFT	Erster Flugversuch
First Full Thrust & Full Burn Time Test	-	FT&FBT	Erster Versuch mit max. Schub & Brenndauer
First Full Thrust Test	-	FTT	Erster Versuch mit max. Schub
Fuel Mass Flow Rate	kg/s	mpunktF	Brennstoffmassenstrom
Full Power Level	kW	Pmax	maximale Leistung
Gimbal Angle Maximum	¡	alphagimbal	Steuerungsverstellwinkel
Heat Exchanger Mass	kg	mheatex	Wörmetauschermasse
Ignition Test/First Hot Firing	-	FIT	Erster Zündversuch
Impulse Bit Minimum	s	tIbit,min	Impulse Bit
Last Flight	-	LF	Letzter Flug
Main Designer	-	fdes	Gestaltungsorganisation
Manifold Mass	kg	mman	Masse des Verteilerrohrs
Manufacturer	-	manu	Fertigungsorganisation
Mass Flow Rate	kg/s	mpunkt	Gesamtmassenstrom
Mixture Ratio	-	O_F	Mischungsverhältnis der TS
Mixture Ratio Maximum	-	O_Fmax	maximales Mischungsverhältnis der TS
Mixture Ratio Minimum	-	O_Fmin	minimales Mischungsverhältnis der TS
Nozzle Area Ratio	-	Ae_At	Düsenflächenverhältnis
Nozzle Area Ratio End	-	Ae_At,e	Düsenflächenverhältnis nach Betrieb
Nozzle Exit Area	m^2	Ae	Düsenendfläche
Nozzle Exit Diameter	m	nde	Düsenenddurchmesser
Nozzle Exit Pressure	pa	pe	Düsenenddruck
Nozzle Expansion Ratio	-	pc_pe	Düsenexpansionsverhältnis
Nozzle Length	m	nl	Düsenlänge
Nozzle Mass	kg	mnoz	Düsenmasse
Nozzle Throat Area	m^2	At	Düsenhalsfläche

Nozzle Throat Diameter	m	ndt	Düsenhalsdurchmesser
Nozzle Throat Pressure	pa	pt	Nozzle Throat Pressure
Nozzle Throat Velocity	m/s	vt	Nozzle Throat Velocity
Number Of Restarts	-	nrestart	Wiederzündbarkeit
Oxidizer Mass Flow Rate	kg/s	mpunktox	Massenstrom des Oxidators
PCCS Mass	kg	mpccs	Masse des Stromregelsystems
Prelimenary Design End	-	Phase Be	Ende der Vorentwurfsphase
Prelimenary Design Start	-	Phase B	Beginn der Vorentwurfsphase
Production End	-	Phase Ee	Produktionsende
Production Start	-	Phase E	Produktionsbeginn
Program/Project Definition End	-	Phase Ce	Ende der Entwurfsphase
Program/Project Definition Start	-	Phase C	Beginn der Entwurfsphase
Proof Pressure	pa	pproof	Prüfdruck
Pulse Duration	s	tlbit	Pulsdauer
Qualification Test End	-	QTE	Ende der Qualifikationstests
Response Time,10 % Thrust	s	ton,thrust,10%	Dauer des Schubaufbaus bei 10 % Schub
Response Time,90 % Thrust	s	ton,thrust,90%	Dauer des Schubaufbaus bei 90 % Schub
Specific Impulse	s	Isp	Spezifische Impuls
Specific Impulse Sea Level	m/s	Isp,SL	Spezifische Impuls (Meereshöhe)
Specific Impulse Vacuum	m/s	Isp,Vac	Spezifische Impuls im Vakuum
Specific Mass Flow Rate	kg/(s m^2)	mpunkt_At	Spezifischer Massenstrom
Specific Power	kW	Psp	Spezifische Leistung
Throttleability	-	Throttl	Drosselung bzw. Regelung
Throttling Maximum	%	Throttl,max	maximale Drosselung
Throttling Minimum	%	Throttl,min	Minimale Drosselung
Thrust	N	F	Schub
Thrust Coefficient	-	cF	Thrust Coefficient
Thrust Coefficient Sea Level	m/s	cFsl	Thrust Coefficient Sea Level
Thrust Coefficient Vacuum	m/s	cFvac	Thrust Coefficient Vacuum
Thrust Maximum	N	Fmax	Maximalschub
Thrust Minimum	N	Fmin	Minimalschub
Thrust Sea Level	N	Fsl	Schub auf Meereshöhe
Thrust Vacuum	N	Fvac	Vakuumschub
Thrust Vacuum Max	N	Fmax,Vac	Maximaler Vakuumschub
Thrust Vacuum Min	N	Fmin,Vac	Minimaler Vakuumschub
Thruster Manufacturer	-	Thrmanu	Fertigungsorganisation der Triebwerke
Time Off	s	toff	Abschaltdauer
Time On	s	ton	Anfahrdauer
Total Efficiency	-	nütot	Gesamtwirkungsgrad
Total Mass	kg	mtot	Gesamtmasse
Treibstoffdichte	kg/m^2	rohp	Treibstoffdichte
Tube Mass	kg	mtube	Masse der Leitungen
Van den Kerkhove-Faktor	-	Gamma	Van den Kerkhove-Faktor
Webb-Thickness	m	b	Webb-Thickness
Wet Mass	kg	mwet	Startmasse
Zündverzugszeit	s	deltaZ	Zündverzugszeit
halber Öffnungswinkel	¡	alpha	halber Öffnungswinkel
spezielle Gaskonstante	J/kg*K	R	spezielle Gaskonstante
Äussere Klemmung	-	Ka	Äussere Klemmung

10.3 Regeln

Launcher

Liefert Attribut	ID	Regel	Benötigt Attribut(e)	
Abolishment Cost	613	CA = LCC-(CD+(CF+CL)*Nträger+COPS)	Number of Launcher	Nträger
			Operation Cost	COPS
			Manufacturing Cost	CF
			Development Cost	CD
			Launch Cost	CL
			Life Cycle Cost	LCC
Average Launch Rate	708	nstart = Mittelwert der eingetragenen Starts		
Baufaktor	720	Mnet_M8 = Mnet/M8	Net Mass	Mnet
			Usable Propellant Mass	M8
Burn-Out Mass	697	Mc = Mnet	Net Mass	Mnet
Coarse Structure Mass	694	Ms = M2+M3+M4+M5+M6	Propellant Residuals Mass	M6
			Propulsion System Mass	M4
			Guidance and Control Mass	M2
			Recovery System Mass	M5
			Structure Mass	M3
	695	Ms = Ms_M0 *M0	Lift-Off Mass	M0
			Structure Ratio	Ms_M0
	696	Ms = M0 / M0_Ms	Lift-Off Mass	M0
			Growth Factor	M0_Ms
Conceptual Design End	626	Launcher Conceptual Design End muss größer sein als Launcher Prelimenary Design Start	Prelimenary Design Start	Phase B
	627	Launcher Conceptual Design End muss kleiner sein als Launcher Conceptual Design Start	Conceptual Design Start	Phase A
Conceptual Design Start	625	Launcher Conceptual Design Start muss größer sein als Launcher Conceptual Design End	Conceptual Design End	Phase Ae
Customer	722	Customer = Satellitenbetreiber		
DeltavGEO	719	dvGEO = 10750		
DeltavLEO	718	dvLEO = 7909		
Development Manager	723	Main Developer = Raumfahrtbehörde d.Startnation		
Development Cost	614	CD = LCC-(CF+CL)*Nträger-COPS-CA	Number of Launcher	Nträger
			Operation Cost	COPS
			Abolishment Cost	CA
			Launch Cost	CL
			Life Cycle Cost	LCC
			Manufacturing Cost	CF
Development End	638	Launcher Development End muss größer sein als Launcher Final Design Review	Final Design Review	FDR
	639	Launcher Development End muss kleiner sein als Launcher Development Start	Development Start	Phase D
Development Start	636	Launcher Development Start muss größer sein als Launcher Development End	Development End	Phase De
	637	Launcher Development Start muss kleiner sein als Launcher Program/Project Definition End	Program/Project Definition End	Phase Ce
Direct Operation Cost	615	DOC = COPS-IOC-VAC-MRC-LSC	Indirect Operation Cost	IOC
			Launch Site and Range Cost	LSC
			Operation Cost	COPS
			Vehicle Amortization Cost	VAC
			Maintenance and Refurbishment Cost	MRC
Dry Mass	682	Mdry = Summe aller Stufenmassen	Dry Mass	Mdry
	683	Mdry = M3+M4+M5	Recovery System Mass	M5
			Structure Mass	M3
			Propulsion System Mass	M4
	684	Mdry = M2+M3+M4+M5	Propulsion System Mass	M4
			Recovery System Mass	M5
			Structure Mass	M3
			Guidance and Control Mass	M2
	685	Mdry = M0-M6-M7-M8	Lift-Off Mass	M0
			Propellant Residuals Mass	M6
			Usable Extra Propellant Mass	M7
			Usable Propellant Mass	M8
	686	Mdry = Mnet-M7	Usable Extra Propellant Mass	M7
			Net Mass	Mnet
	687	Mdry = Ms - M6 - M7	Usable Extra Propellant Mass	M7
			Propellant Residuals Mass	M6
			Coarse Structure Mass	Ms
Final Design Review	640	Launcher Final Design Review muss größer sein als Launcher Production Start	Production Start	Phase E
	641	Launcher Final Design Review muss kleiner sein als Launcher Development End	Development End	Phase De
Final Documentation/Report	652	Launcher Final Documentation/Report muss kleiner sein als Launcher Last Flight	Last Flight	LF
First Flight Attempt	644	Launcher First Flight Attempt muss größer sein als Launcher Succesful Flight Demonstration	Succesful Flight Demonstration	SFD
	645	Launcher First Flight Attempt muss kleiner sein als Launcher Production Start	Production Start	Phase E
Gross Factor	709	M0_M1 = M0/M1	Payload Mass	M1
			Lift-Off Mass	M0

	710	M0_M1 > = 18		
Growth Factor	711	M0_Ms = M0/Mdry	Lift-Off Mass	M0
			Dry Mass	Mdry
Guidance and Control Mass	655	M2 = Summe aller Stufenmassen	Guidance and Control Mass	M2
	656	M2 = M0-M3-M4-M5-M6-M7-M8	Lift-Off Mass	M0
			Recovery System Mass	M5
			Structure Mass	M3
			Usable Extra Propellant Mass	M7
			Usable Propellant Mass	M8
			Propellant Residuals Mass	M6
			Propulsion System Mass	M4
	657	M2 = Ms-M3-M4-M5-M6	Propellant Residuals Mass	M6
			Propulsion System Mass	M4
			Recovery System Mass	M5
			Structure Mass	M3
			Coarse Structure Mass	Ms
Indirect Operation Cost	616	IOC = COPS-VAC-MRC-LSC-DOC	Vehicle Amortization Cost	VAC
			Operation Cost	COPS
			Direct Operation Cost	DOC
			Launch Site and Range Cost	LSC
			Maintenance and Refurbishment Cost	MRC
Last Flight	650	Launcher Last Flight muss größer sein als Launcher Final Documentation/Report	Final Documentation/Report	FR
	651	Launcher Last Flight muss kleiner sein als Launcher Production End	Production End	Phase Ee
Launch Cost	617	CL = (LCC-CD-COPS-CA)/Nträger-CF	Manufacturing Cost	CF
			Number of Launcher	Nträger
			Operation Cost	COPS
			Life Cycle Cost	LCC
			Development Cost	CD
			Abolishment Cost	CA
Launch Service Agency	724	Launch Service Provider = Vehicle Operator	Vehicle Operator	operat
Launch Site and Range Cost	618	LSC = COPS-VAC-MRC-IOC-DOC	Vehicle Amortization Cost	VAC
			Maintenance and Refurbishment Cost	MRC
			Direct Operation Cost	DOC
			Indirect Operation Cost	IOC
			Operation Cost	COPS
Life Cycle Cost	619	LCC = CD+(CF+CL)*Nträger+COPS+CA	Number of Launcher	Nträger
			Operation Cost	COPS
			Launch Cost	CL
			Development Cost	CD
			Manufacturing Cost	CF
			Abolishment Cost	CA
Lift-Off Mass	698	M0 = Summe aller Stufenmassen	Lift-Off Mass	M0
	699	M0 = M2+M3+M4+M5+M6+M7+M8	Guidance and Control Mass	M2
			Recovery System Mass	M5
			Structure Mass	M3
			Propellant Residuals Mass	M6
			Propulsion System Mass	M4
			Usable Extra Propellant Mass	M7
			Usable Propellant Mass	M8
	700	M0 = M3+M4+M5+M6+M7+M8	Usable Extra Propellant Mass	M7
			Usable Propellant Mass	M8
			Propellant Residuals Mass	M6
			Propulsion System Mass	M4
			Recovery System Mass	M5
			Structure Mass	M3
	701	M0 = Mdry+M6+M7+M8	Dry Mass	Mdry
			Propellant Residuals Mass	M6
			Usable Extra Propellant Mass	M7
			Usable Propellant Mass	M8
	702	M0 = Mnet+M8	Usable Propellant Mass	M8
			Net Mass	Mnet
	703	M0 = M0_Ms * Ms	Coarse Structure Mass	Ms
			Growth Factor	M0_Ms
	704	M0 = Mdry+M8	Dry Mass	Mdry
			Usable Propellant Mass	M8
	705	M0 = Mdry / Ms_M0	Dry Mass	Mdry
			Structure Ratio	Ms_M0
	706	M0 = M8 / M8_M0	Usable Propellant Mass	M8
			Propellant Ratio	M8_M0
	707	M0 = R * M8/ (R-1)	Usable Propellant Mass	M8
			Overall Mass Ratio	R

Maintenance and Refurbishment Cost	620	MRC = COPS-VAC-IOC-LSC-DOC	Operation Cost	COPS
			Direct Operation Cost	DOC
			Indirect Operation Cost	IOC
			Launch Site and Range Cost	LSC
			Vehicle Amortization Cost	VAC
Manufacturing Cost	621	CF = (LCC-CD-COPS-CA)/Nträger-CL	Abolishment Cost	CA
			Launch Cost	CL
			Life Cycle Cost	LCC
			Development Cost	CD
			Number of Launcher	Nträger
			Operation Cost	COPS
Maximum Diameter	653	Maximale Durchmesser aller Stufen ohne Parallelstufen	Stage Diameter Max	
Net Mass	688	Mnet = Summe aller Stufenmassen	Net Mass	Mnet
	689	Mnet = M3+M4+M5+M6+M7	Recovery System Mass	M5
			Structure Mass	M3
			Usable Extra Propellant Mass	M7
			Propellant Residuals Mass	M6
			Propulsion System Mass	M4
	690	Mnet = M2+M3+M4+M5+M6+M7	Propellant Residuals Mass	M6
			Propulsion System Mass	M4
			Usable Extra Propellant Mass	M7
			Recovery System Mass	M5
			Structure Mass	M3
			Guidance and Control Mass	M2
	691	Mnet = Mdry+M6+M7	Dry Mass	Mdry
			Usable Extra Propellant Mass	M7
			Propellant Residuals Mass	M6
	692	Mnet = M0-M8	Usable Propellant Mass	M8
			Lift-Off Mass	M0
	693	Mnet = Mnet_M8 * M8	Usable Propellant Mass	M8
			Baufaktor	Mnet_M8
Number Of Boosters	600	Anzahl der Einträge in der Tabelle LAUNCHER_Stage mit Stage_No = 0	Stage_No	
Number Of Stages	601	Anzahl der Einträge in der Tabelle LAUNCHER_Stage	Launcher_Stage	
Number of Launchers	612	Nträger = (LCC-CD-COPS-CA)/(CF+CL)	Launch Cost	CL
			Life Cycle Cost	LCC
			Development Cost	CD
			Operation Cost	COPS
			Abolishment Cost	CA
			Manufacturing Cost	CF
Operation Cost	622	COPS = DOC+IOC+VAC+MRC+LSC	Vehicle Amortization Cost	VAC
			Direct Operation Cost	DOC
			Indirect Operation Cost	IOC
			Launch Site and Range Cost	LSC
			Maintenance and Refurbishment Cost	MRC
	623	COPS = LCC-CD-(CF+CL)*Nträger-CA	Launch Cost	CL
			Life Cycle Cost	LCC
			Development Cost	CD
			Number of Launcher	Nträger
			Manufacturing Cost	CF
			Abolishment Cost	CA
Overall Length	654	ltot = Summe der Länge aller Stufen	Stage Length	
Overall Mass Ratio	715	R = M0/(M0-M8)	Lift-Off Mass	M0
			Usable Propellant Mass	M8
	716	R = exp (dvLEO/ce)	ce	
			dvLEO	
			e	
	717	R = exp (dvGEO/ce)	ce	
			dvGEO	
			e	
Payload Ratio	712	M1_M0 = M1/M0	Payload Mass	M1
			Lift-Off Mass	M0
Prelimenary Design End	630	Launcher Prelimenary Design End muss größer sein als Launcher Program/Project Definition Start	Program/Project Definition Start	Phase C
	631	Launcher Prelimenary Design End muss kleiner sein als Launcher Prelimenary Design Start	Prelimenary Design Start	Phase B
Prelimenary Design Start	628	Launcher Prelimenary Design Start muss größer sein als Launcher Prelimenary Design End	Prelimenary Design End	Phase Be
	629	Launcher Prelimenary Design Start muss kleiner sein als Launcher Conceptual Design End	Conceptual Design End	Phase Ae
Production End	648	Launcher Production End muss größer sein als Launcher Last Flight	Last Flight	LF
	649	Launcher Production End muss kleiner sein als Launcher Succesful Flight Demonstration	Succesful Flight Demonstration	SFD
Production Start	642	Launcher Production Start muss größer sein als Launcher First Flight Attempt	First Flight Attempt	FFA
	643	Launcher Production Start muss kleiner sein als Launcher Final Design Review	Final Design Review	FDR

Program/Project Definition End	634	Launcher Program/Project Definition End muss größer sein als Launcher Development Start	Development Start	Phase D
	635	Launcher Program/Project Definition End muss kleiner sein als Launcher Program/Project Definition Start	Program/Project Definition Start	Phase C
Program/Project Definition Start	632	Launcher Program/Project Definition Start muss größer sein als Launcher Program/Project Definition End	Program/Project Definition End	Phase Ce
	633	Launcher Program/Project Definition Start muss kleiner sein als Launcher Prelimenary Design End	Prelimenary Design End	Phase Be
Propellant Ratio	714	M8_M0 = M8/M0	Usable Propellant Mass	M8
			Lift-Off Mass	M0
Propellant Residuals Mass	673	M6 = Summe aller Stufenmassen	Propellant Residuals Mass	M6
	674	M6 = M0-M2-M3-M4-M5-M7-M8	Propulsion System Mass	M4
			Usable Extra Propellant Mass	M7
			Usable Propellant Mass	M8
			Lift-Off Mass	M0
			Guidance and Control Mass	M2
			Recovery System Mass	M5
			Structure Mass	M3
	675	M6 = Ms-M2-M3-M4-M5	Recovery System Mass	M5
			Structure Mass	M3
			Guidance and Control Mass	M2
			Propulsion System Mass	M4
			Coarse Structure Mass	Ms
	676	M6 = Mnet-M3-M4-M5-M7	Net Mass	Mnet
			Propellant Residuals Mass	M6
			Propulsion System Mass	M4
			Usable Extra Propellant Mass	M7
			Recovery System Mass	M5
Propulsion System Mass	663	M4 = Summe aller Stufenmassen	Propulsion System Mass	M4
	664	M4 = M0-M2-M3-M5-M6-M7-M8	Propellant Residuals Mass	M6
			Usable Extra Propellant Mass	M7
			Usable Propellant Mass	M8
			Recovery System Mass	M5
			Structure Mass	M3
			Guidance and Control Mass	M2
			Lift-Off Mass	M0
	665	M4 = Ms-M2-M3-M5-M6	Guidance and Control Mass	M2
			Recovery System Mass	M5
			Structure Mass	M3
			Propellant Residuals Mass	M6
			Coarse Structure Mass	Ms
	666	M4 = Mdry-M3-M5	Recovery System Mass	M5
			Structure Mass	M3
			Dry Mass	Mdry
	667	M4 = Mnet-M3-M5-M6-M7	Recovery System Mass	M5
			Structure Mass	M3
			Propellant Residuals Mass	M6
			Usable Extra Propellant Mass	M7
			Net Mass	Mnet
Recovery System Mass	668	M5 = Summe aller Stufenmassen	Recovery System Mass	M5
	669	M5 = M0-M2-M3-M4-M6-M7-M8	Structure Mass	M3
			Guidance and Control Mass	M2
			Lift-Off Mass	M0
			Usable Extra Propellant Mass	M7
			Usable Propellant Mass	M8
			Propellant Residuals Mass	M6
			Propulsion System Mass	M4
	670	M5 = Ms-M2-M3-M4-M6	Propellant Residuals Mass	M6
			Propulsion System Mass	M4
			Guidance and Control Mass	M2
			Structure Mass	M3
			Coarse Structure Mass	Ms
	671	M5 = Mdry-M3-M4	Structure Mass	M3
			Dry Mass	Mdry
			Propulsion System Mass	M4
	672	M5 = Mnet-M3-M4-M6-M7	Propellant Residuals Mass	M6
			Propulsion System Mass	M4
			Usable Extra Propellant Mass	M7
			Structure Mass	M3
			Net Mass	Mnet
Reuseability	602	yes wenn Landing_M0de not none	Landing Mode	modeland
	603	yes wenn Landing_M0de = HL oder VL	Landing Mode	modeland
	604	no wenn Landing_M0de leer	Landing Mode	modeland
	605	yes wenn alle Stufen reuseable	Stage_Reuseability	
	606	partial wenn nicht alle Stufen wiederverwendbar	Stage_Reuseability	
	607	no wenn keine Stufe wiederverwendbar	Stage_Reuseability	

Status	608	aktuelles Datum mit Phasendaten vergl.:	Program/Project Definition Start	Phase C
			Prelimenary Design Start	Phase B
			Conceptual Design Start	Phase A
	609	wenn D, = under development	Development Start	Phase D
	610	wenn E, = operational	Production Start	Phase E
	611	wenn FR, = no more available	Final Documentation/Report	FR
Structure Mass	658	M3 = Summe aller Stufenmassen	Structure Mass	M3
	659	M3 = M0-M2-M4-M5-M6-M7-M8	Recovery System Mass	M5
			Guidance and Control Mass	M2
			Lift-Off Mass	M0
			Propellant Residuals Mass	M6
			Propulsion System Mass	M4
			Usable Extra Propellant Mass	M7
			Usable Propellant Mass	M8
	660	M3 = Ms-M2-M4-M5-M6	Propellant Residuals Mass	M6
			Propulsion System Mass	M4
			Guidance and Control Mass	M2
			Recovery System Mass	M5
			Coarse Structure Mass	Ms
	661	M3 = Mdry-M4-M5	Recovery System Mass	M5
			Dry Mass	Mdry
			Propulsion System Mass	M4
	662	M3 = Mnet-M4-M5-M6-M7	Propellant Residuals Mass	M6
			Propulsion System Mass	M4
			Usable Extra Propellant Mass	M7
			Recovery System Mass	M5
			Net Mass	Mnet
Structure Ratio	713	Ms_M0 = Mdry/M0	Dry Mass	Mdry
			Lift-Off Mass	M0
Succesful Flight Demonstration	646	Launcher Succesful Flight Demonstration muss größer sein als Launcher Production End	Production End	Phase Ee
	647	Launcher Succesful Flight Demonstration muss kleiner sein als Launcher First Flight Attempt	First Flight Attempt	FFA
Usable Extra Propellant Mass	677	M7 = Summe aller Stufenmassen	Usable Extra Propellant Mass	M7
	678	M7 = M0-M2-M3-M4-M5-M6-M8	Usable Propellant Mass	M8
			Propellant Residuals Mass	M6
			Propulsion System Mass	M4
			Guidance and Control Mass	M2
			Lift-Off Mass	M0
			Recovery System Mass	M5
			Structure Mass	M3
	679	M7 = Mnet-M3-M4-M5-M6	Recovery System Mass	M5
			Structure Mass	M3
			Propellant Residuals Mass	M6
			Propulsion System Mass	M4
			Net Mass	Mnet
Usable Propellant Mass	680	M8 = Summe aller Stufenmassen	Usable Propellant Mass	M8
	681	M8 = M0-M2-M3-M4-M5-M6-M7	Usable Extra Propellant Mass	M7
			Propellant Residuals Mass	M6
			Propulsion System Mass	M4
			Recovery System Mass	M5
			Structure Mass	M3
			Lift-Off Mass	M0
			Guidance and Control Mass	M2
Vehicle Amortization Cost	624	VAC = COPS-IOC-MRC-LSC-DOC	Indirect Operation Cost	IOC
			Maintenance and Refurbishment Cost	MRC
			Launch Site and Range Cost	LSC
			Direct Operation Cost	DOC
			Operation Cost	COPS

Stage

Liefert Attribut	ID	Regel	Benötigt Attribut(e)	
Actuator System Mass	444	mact = mdry - Equipment Bay Mass - Interstage Adapter Mass - Propulsion System Mass	Equipment Bay Mass	mVEB
			Propulsion System Mass	m4
			Dry Mass	mdry
			Interstage Adapter Mass	mISA
Baufaktor	502	mnet_m8 = mnet/m8	Usable Propellant Mass	m8
			Net Mass	mnet
Burn-Out Mass	445	mc = mnet	Net Mass	mnet
Conceptual Design End	402	Prelimenary Design Start muss größer sein als Conceptual Design End	Prelimenary Design Start	Phase B
	403	Conceptual Design Start muss kleiner sein als Conceptual Design End	Conceptual Design Start	Phase A
Conceptual Design Start	401	Conceptual Design End muss größer sein als Conceptual Design Start	Conceptual Design End	Phase Ae
Development End	414	Final Design Review muss größer sein als Development End	Final Design Review	FBTT
	415	Development Start muss kleiner sein als Development End	Development Start	Phase D
Development Start	412	Development End muss größer sein als Development Start	Development End	Phase De
	413	Program/Project Definition End muss kleiner sein als Development Start	Program/Project Definition End	Phase Ce
Dry Mass	446	mdry = m3+m4+m5	Propulsion System Mass	m4
			Structure Mass	m3
			Recovery System Mass	m5
	447	mdry = m2+m3+m4+m5	Propulsion System Mass	m4
			Recovery System Mass	m5
			Structure Mass	m3
			Guidance and Control Mass	m2
	448	mdry = mdry=m0-m6-m7-m8	Lift-Off Mass	m0
			Usable Extra Propellant Mass	m7
			Usable Propellant Mass	m8
			Propellant Residuals Mass	m6
	449	mdry = mnet-m7	Usable Extra Propellant Mass	m7
			Net Mass	mnet
	450	mdry = mo-m8tot	Propellant Total Mass	m8tot
			Lift-Off Mass	m0
Equipment Bay Diameter	433	dVEB = Diameter letzte Stufe	Stage Diameter Max	dstage
Equipment Bay Mass	451	mVEB = mdry - Actuator System Mass - Interstage Adapter Mass - Propulsion System Mass	Propulsion System Mass	m4
			Dry Mass	mdry
			Interstage Adapter Mass	mISA
			Actuator System Mass	mact
Final Burn Time Test	420	Production Start muss größer sein als Final Burn Time Test	Production Start	Phase E
	421	Ignition Test/First Hot Firing muss kleiner sein als Final Burn Time Test	Ignition Test/First Hot Firing	
Final Design Review	416	Ignition Test/First Hot Firing muss größer sein als Final Design Review	Ignition Test/First Hot Firing	
	417	Development End muss kleiner sein als Final Design Review	Development End	Phase De
Final Documentation/Report	432	Last Flight muss kleiner sein als Final Documentation/Report	Last Flight	LF
First Flight Attempt	424	Succesful Flight Demonstration muss größer sein als First Flight Attempt	Succesful Flight Demonstration	SFD
	425	Production Start muss kleiner sein als First Flight Attempt	Production Start	Phase E
Fuel Tank Diameter	434	dfueltank = Wurzel (4 * mfuel / FuelDichte * lfueltank * pi)	Propellant Fuel Mass	mfuel
			Fuel Density	
			Fuel Tank Length	lfueltank
	435	dfueltank ist größer Wurzel (4 * mfuel / FuelDichte * lfueltank * pi)	Fuel Density	
			Fuel Tank Length	lfueltank
			Propellant Fuel Mass	mfuel
Fuel Tank Length	436	lfueltank = (4 * mfuel / FuelDichte * dfueltank^2 * pi)	Propellant Fuel Mass	mfuel
			Fuel Density	
			Fuel Tank Diameter	dfueltank
	437	lfueltank ist größer (4 * mfuel / FuelDichte * dfueltank^2 * pi)	Fuel Density	
			Fuel Tank Diameter	dfueltank
			Propellant Fuel Mass	mfuel
Fuel Tank Volume	503	Vfueltank = pi/4 * (Fuel Tank Diameter)^2 * Fuel Tank Length	Fuel Tank Diameter	dfueltank
			Fuel Tank Length	lfueltank
	504	Vfueltank = mfuel / Fuel Dichte	Fuel Density	
Groß Mass	452	mgroß = m0	Lift-Off Mass	m0
Growth Factor	505	m0_ms = m0/mdry	Lift-Off Mass	m0
			Dry Mass	mdry
Guidance and Control Mass	453	m2 = m0-m3-m4-m5-m6-m7-m8	Recovery System Mass	m5
			Structure Mass	m3
			Lift-Off Mass	m0
			Propellant Residuals Mass	m6
			Propulsion System Mass	m4
			Usable Extra Propellant Mass	m7
			Usable Propellant Mass	m8
	454	m2 = mdry-m3-m4-m5-m6	Propellant Residuals Mass	m6
			Propulsion System Mass	m4
			Recovery System Mass	m5
			Structure Mass	m3
			Dry Mass	mdry
Ignition Test/First Hot Firing	418	Final Burn Time Test muss größer sein als Ignition Test/First Hot Firing	Final Burn Time Test	FIT
	419	Final Design Review muss kleiner sein als Ignition Test/First Hot Firing	Final Design Review	FBTT

Interstage Adapter Mass	455	mISA = mdry – Actuator System Mass – Equipment Bay Mass – Propulsion System Mass	Dry Mass	mdry
			Propulsion System Mass	m4
			Actuator System Mass	mact
			Equipment Bay Mass	mVEB
Last Flight	430	Final Documentation/Report muss größer sein als Last Flight	Final Documentation/Report	FR
	431	Production End muss kleiner sein als Last Flight	Production End	Phase Ee
Lift-Off Mass	456	m0 = m2+m3+m4+m5+m6+m7+m8	Propellant Residuals Mass	m6
			Propulsion System Mass	m4
			Usable Extra Propellant Mass	m7
			Usable Propellant Mass	m8
			Guidance and Control Mass	m2
			Recovery System Mass	m5
			Structure Mass	m3
	457	m0 = m3+m4+m5+m6+m7+m8	Recovery System Mass	m5
			Structure Mass	m3
			Usable Extra Propellant Mass	m7
			Usable Propellant Mass	m8
			Propellant Residuals Mass	m6
			Propulsion System Mass	m4
	458	m0 = mdry+m6+m7+m8	Propellant Residuals Mass	m6
			Usable Extra Propellant Mass	m7
			Usable Propellant Mass	m8
			Dry Mass	mdry
	459	m0 = mnet+m8	Usable Propellant Mass	m8
			Net Mass	mnet
	460	m0 = m0_ms * mdry	Dry Mass	mdry
			Growth Factor	m0_ms
	461	m0 = mdry+m8	Dry Mass	mdry
			Usable Propellant Mass	m8
	462	m0 = mdry / ms_m0	Dry Mass	mdry
			Structure Ratio	ms_m0
	463	m0 = m8 / m8_m0	Usable Propellant Mass	m8
			Propellant Ratio	m8_m0
	464	m0 = R * m8/ (R-1)	Usable Propellant Mass	m8
			Overall Mass Ratio	R
	465	m0 <= Groß mass	Groß Mass	mgroß
Net Mass	466	mnet = m3+m4+m5+m6+m7	Propulsion System Mass	m4
			Structure Mass	m3
			Recovery System Mass	m5
			Usable Extra Propellant Mass	m7
			Propellant Residuals Mass	m6
	467	mnet = m2+m3+m4+m5+m6+m7	Propellant Residuals Mass	m6
			Usable Extra Propellant Mass	m7
			Propulsion System Mass	m4
			Structure Mass	m3
			Recovery System Mass	m5
			Guidance and Control Mass	m2
	468	mnet = mdry+m6+m7	Dry Mass	mdry
			Usable Extra Propellant Mass	m7
			Propellant Residuals Mass	m6
	469	mnet = m0-m8	Usable Propellant Mass	m8
			Lift-Off Mass	m0
	470	mnet = mnet_m8 * m8	Usable Propellant Mass	m8
			Baufaktor	mnet_m8
Overall Mass Ratio	506	R = m0/(m0-m8)	Usable Propellant Mass	m8
			Lift-Off Mass	m0
Oxidizer Tank Diameter	438	doxtank = Wurzel (4 * mox / OxDichte * loxtank * pi)	Propellant Oxidizer Mass	mox
			Ox Density	
			Oxidizer Tank Length	loxtank
	439	doxtank ist größer Wurzel (4 * mox / OxDichte * loxtank * pi)	Oxidizer Tank Length	loxtank
			Propellant Oxidizer Mass	mox
			Ox Density	
Oxidizer Tank Length	440	loxtank = (4 * mox / OxDichte * doxtank^2 * pi)	Ox Density	
			Propellant Oxidizer Mass	mox
			Oxidizer Tank Diameter	doxtank
	441	loxtank ist größer (4 * mox / OxDichte * doxtank^2 * pi)	Oxidizer Tank Diameter	doxtank
			Propellant Oxidizer Mass	mox
			Ox Density	
Oxidizer Tank Volume	507	Voxtank = pi/4 * (Oxidizer Tank Diameter)^2 * Oxidizer Tank Length	Oxidizer Tank Diameter	doxtank
			Oxidizer Tank Length	loxtank
	508	Voxtank = mox * OxDichte	Ox Density	
			Propellant Oxidizer Mass	mox
Prelimenary Design End	406	Program/Project Definition Start muss größer sein als Prelimenary Design End	Program/Project Definition Start	Phase C
	407	Prelimenary Design Start muss kleiner sein als Prelimenary Design End	Prelimenary Design Start	Phase B
Prelimenary Design Start	404	Prelimenary Design End muss größer sein als Prelimenary Design Start	Prelimenary Design End	Phase Be
	405	Conceptual Design End muss kleiner sein als Prelimenary Design Start	Conceptual Design End	Phase Ae

Pressurization Gas Mass	471	mgas = m8tot – mfuel – msonst+ mox	Propellant Fuel Mass	mfuel
			Propellant Oxidizer Mass	mox
			Propellant Other Mass	msonst
			Propellant Total Mass	m8tot
	472	mgas = mpunktF*Anzahl Triebwerke * tc	Number of engines	
			Fuel Mass Flow Rate	
			Burn Time	
	473	mgas = mpunktOx*Anzahl Triebwerke * tc	Burn Time	
			Number Of Engines	Noe
			Oxidizer Mass Flow Rate	
Production End	428	Last Flight muss größer sein als Production End	Last Flight	LF
	429	Production Start muss kleiner sein als Production End	Production Start	Phase E
Production Start	422	First Flight Attempt muss größer sein als Production Start	First Flight Attempt	FFA
	423	Final Burn Time Test muss kleiner sein als Production Start	Final Burn Time Test	FIT
Program/Project Definition End	410	Development Start muss größer sein als Program/Project Definition End	Development Start	Phase D
	411	Program/Project Definition Start muss kleiner sein als Program/Project Definition End	Program/Project Definition Start	Phase C
Program/Project Definition Start	408	Program/Project Definition End muss größer sein als Program/Project Definition Start	Program/Project Definition End	Phase Ce
	409	Prelimenary Design End muss kleiner sein als Program/Project Definition Start	Prelimenary Design End	Phase Be
Propellant Fuel Mass	66	mfuel = m8tot – msonst – mox – Pressurization Gas Mass	Pressurization Gas Mass	mgas
			Propellant Other Mass	msonst
			Propellant Oxidizer Mass	mox
			Propellant Total Mass	m8tot
Propellant Other Mass	68	msonst = m8tot – mfuel – mox – Pressurization Gas Mass	Propellant Total Mass	m8tot
			Propellant Fuel Mass	mfuel
			Propellant Oxidizer Mass	mox
			Pressurization Gas Mass	mgas
Propellant Oxidizer Mass	67	mox = m8tot – mfuel – msonst – Pressurization Gas Mass	Pressurization Gas Mass	mgas
			Propellant Fuel Mass	mfuel
			Propellant Total Mass	m8tot
			Propellant Other Mass	msonst
Propellant Ratio	509	m8_m0 = m8/m0	Usable Propellant Mass	m8
			Lift-Off Mass	m0
Propellant Residuals Mass	474	m6 = m0-m2-m3-m4-m5-m7-m8	Lift-Off Mass	m0
			Guidance and Control Mass	m2
			Structure Mass	m3
			Usable Extra Propellant Mass	m7
			Usable Propellant Mass	m8
			Propulsion System Mass	m4
			Recovery System Mass	m5
			Propellant Residuals Mass	m6
	475	m6 = mdry-m2-m3-m4-m5	Propulsion System Mass	m4
			Recovery System Mass	m5
			Structure Mass	m3
			Dry Mass	mdry
			Guidance and Control Mass	m2
	476	m6 = mnet-m3-m4-m5-m7	Structure Mass	m3
			Propulsion System Mass	m4
			Recovery System Mass	m5
			Usable Extra Propellant Mass	m7
			Net Mass	mnet
	477	m6 = m8	Usable Propellant Mass	m8
	478	m6 = mfuel + mox	Propellant Fuel Mass	mfuel
			Propellant Oxidizer Mass	mox
	479	m6 = mpunkt * tc * Anzahl Triebwerke	Number of Engines	NOE
			Mass Flow Rate	mpunkt
			Burn Time	tc
Propellant Total Mass	65	m8tot = mfuel + msonst + mox + Pressurization Gas Mass	Pressurization Gas Mass	mgas
			Propellant Fuel Mass	mfuel
			Propellant Oxidizer Mass	mox
			Propellant Other Mass	msonst
Propulsion System Mass	480	m4 = m0-m2-m3-m5-m6-m7-m8	Propellant Residuals Mass	m6
			Usable Extra Propellant Mass	m7
			Usable Propellant Mass	m8
			Recovery System Mass	m5
			Structure Mass	m3
			Guidance and Control Mass	m2
			Lift-Off Mass	m0
	481	m4 = mdry-m2-m3-m5-m6	Guidance and Control Mass	m2
			Structure Mass	m3
			Dry Mass	mdry
			Recovery System Mass	m5
			Propellant Residuals Mass	m6
	482	m4 = mdry-m3-m5	Recovery System Mass	m5
			Dry Mass	mdry
			Structure Mass	m3

Propulsion System Mass	483	m4 = mnet-m3-m5-m6-m7	Structure Mass	m3
			Recovery System Mass	m5
			Usable Extra Propellant Mass	m7
			Propellant Residuals Mass	m6
			Net Mass	mnet
	484	m4 = mdry – Actuator System Mass – Equipment Bay Mass – Interstage Adapter Mass	Actuator System Mass	mact
			Equipment Bay Mass	mVEB
			Number of engines	
			Dry Mass	mdry
			Interstage Adapter Mass	mISA
	485	m4 = NOE * mtot	Total Mass	
Recovery System Mass	486	m5 = m0-m2-m3-m4-m6-m7-m8	Usable Extra Propellant Mass	m7
			Usable Propellant Mass	m8
			Propulsion System Mass	m4
			Propellant Residuals Mass	m6
			Lift-Off Mass	m0
			Guidance and Control Mass	m2
			Structure Mass	m3
	487	m5 = mdry-m2-m3-m4-m6	Structure Mass	m3
			Dry Mass	mdry
			Guidance and Control Mass	m2
			Propellant Residuals Mass	m6
			Propulsion System Mass	m4
	489	m5 = mdry-m3-m4	Propulsion System Mass	m4
			Dry Mass	mdry
			Structure Mass	m3
	490	m5 = mnet-m3-m4-m6-m7	Structure Mass	m3
			Propulsion System Mass	m4
			Usable Extra Propellant Mass	m7
			Propellant Residuals Mass	m6
			Net Mass	mnet
Stage Accelaration Max	510	amax = Fstage / mc	Burn-Out Mass	mc
			Stage Thrust	Fstage
Stage Diameter Max	442	dstage >= grösster Tankdurchmesser	Tank Diameter	dtank
Stage Thrust	511	Fstage = NOE * F	Thrust	
			Number of engines	
	512	Fstage = NOE * Fvac	Number of engines	
			Thrust Vacuum	
Structure Mass	491	m3 = m0-m2-m4-m5-m6-m7-m8	Usable Extra Propellant Mass	m7
			Usable Propellant Mass	m8
			Propulsion System Mass	m4
			Recovery System Mass	m5
			Propellant Residuals Mass	m6
			Guidance and Control Mass	m2
			Lift-Off Mass	m0
	492	m3 = mdry-m2-m4-m5-m6	Guidance and Control Mass	m2
			Dry Mass	mdry
			Propellant Residuals Mass	m6
			Propulsion System Mass	m4
			Recovery System Mass	m5
	493	m3 = mdry-m4-m5	Propulsion System Mass	m4
			Recovery System Mass	m5
			Dry Mass	mdry
	494	m3 = mnet-m4-m5-m6-m7	Propulsion System Mass	m4
			Recovery System Mass	m5
			Usable Extra Propellant Mass	m7
			Propellant Residuals Mass	m6
			Net Mass	mnet
	495	m3 = ms_m0 * m0	Structure Ratio	ms_m0
			Lift-Off Mass	m0
	496	m3 = m0 / m0_ms	Lift-Off Mass	m0
			Growth Factor	m0_ms
	497	m3 = mdry- Actuator System Mass + Equipment Bay Mass + Interstage Adapter Mass + Propulsion System Mass	Equipment Bay Mass	mVEB
			Interstage Adapter Mass	mISA
			Dry Mass	mdry
			Propulsion System Mass	m4
			Actuator System Mass	mact
Structure Ratio	513	ms_m0 = mdry/m0	Dry Mass	mdry
			Lift-Off Mass	m0
Succesful Flight Demonstration	426	Production End muss größer sein als Succesful Flight Demonstration	Production End	Phase Ee
	427	First Flight Attempt muss kleiner sein als Succesful Flight Demonstration	First Flight Attempt	FFA
Tank Diameter	443	dtank <= Stufendurchmesser	Stage Diameter Max	dstage

Usable Extra Propellant Mass	498	m7 = m0-m2-m3-m4-m5-m6-m8	Propulsion System Mass	m4
			Recovery System Mass	m5
			Usable Propellant Mass	m8
			Propellant Residuals Mass	m6
			Guidance and Control Mass	m2
			Lift-Off Mass	m0
			Structure Mass	m3
	499	m7 = mnet-m3-m4-m5-m6	Structure Mass	m3
			Propellant Residuals Mass	m6
			Propulsion System Mass	m4
			Recovery System Mass	m5
			Net Mass	mnet
Usable Propellant Mass	500	m8 = m0-m2-m3-m4-m5-m6-m7	Propulsion System Mass	m4
			Recovery System Mass	m5
			Usable Extra Propellant Mass	m7
			Propellant Residuals Mass	m6
			Structure Mass	m3
			Lift-Off Mass	m0
			Guidance and Control Mass	m2
	501	m8 = mpunkt *ZA	Mass Flow Rate	
			Aktionszeit	

Engine

Liefert Attribut	ID	Regel	Benötigt Attribut(e)	
Abbrandgeschwindigkeit	53	rpunkt = mpunkt/ (Ap*rohp)	Abbrandoberfläche Mass Flow Rate Treibstoffdichte	Ap mpunkt rohp
	57	rpunkt = b/ tc	Burn Time Webb-Thickness	tc b
Abbrandoberfläche	52	Ap = Ka*At	Äussere Klemmung Nozzle Throat Area	Ka At
	55	Ap = mpunkt / (rpunkt*rohp)	Abbrandgeschwindigkeit Mass Flow Rate	rpunkt mpunkt
Adiabatenexponent	319	Kappa = cp/cv	Spezifische Wärmekapazität (Cp) Spezifische Wärmekapazität (Cv)	
Aktionszeit	233	Za = 2,25		
	234	Za = 37,5		
	235	Za = 110		
Burn Time	58	tc = b / rpunkt	Webb-Thickness Abbrandgeschwindigkeit	b rpunkt
	60	tc = ist größer ZA	Aktionszeit	Za
	61	tc = ZA	Aktionszeit	Za
	163	mtube = m8 / mpunkt * NOE	Number of engines Mass Flow Rate Usable Propellant Mass	 mpunkt
Burn Time Maximum	164	tc,max = tc	Burn Time	tc
Chamber Combustion Pressure	41	mpunkt_At = (2/Kappa+1) * exp(Kappa/1-Kappa)	Adiabatenexponent Nozzle Throat Pressure	Kappa pt
	165	pc = cstern*mpunkt/At	Nozzle Throat Area Mass Flow Rate Characteristic Velocity	At mpunkt cstern
	166	pc = pc_pe*pe	Expansion Ratio Nozzle Exit Pressure	 pe
	167	pc = F /(At*cF)	Nozzle Throat Area Thrust Thrust Coefficient	At F cF
	238	pc = 5000000		
	239	pc = 7000000		
	240	pc = 8500000		
	272	CTPB 10-14: = 50,0	Engine Propellant	
	273	CTPB-type: = 45,4	Engine Propellant	
	274	F2/NH3: = 118,0	Engine Propellant	
	275	GO2/GH2: = 27,6	Engine Propellant	
	276	GOX/synthetic kerosene: = 87,5	Engine Propellant	
	277	H2O2/pentaboran: = 147,0	Engine Propellant	
	278	HEF-20: = 36,3	Engine Propellant	
	279	HNO3/N2H4: = 24,0	Engine Propellant	
	280	HNO3/UDMH: = 67,6	Engine Propellant	
	281	HNO3/amine based: = 43,5	Engine Propellant	
	282	HNO3/kerosene: = 34,5	Engine Propellant	
	283	HTPB-Al: = 48,6	Engine Propellant	
	284	HTPB-Al class 1,3: = 55,9	Engine Propellant	
	285	HTPB-type: = 46,6	Engine Propellant	
	286	IRFNA/0,65Aniline+0,35Furfuryl Alcohol: = 22,3	Engine Propellant	
	287	IRFNA/UDMH: = 35,9	Engine Propellant	
	288	LF2/LNH3: = 118,0	Engine Propellant	
	289	LOX/CH4: = 180,9	Engine Propellant	
	290	LOX/GH2: = 20,7	Engine Propellant	
	291	LOX/LCH4: = 105,0	Engine Propellant	
	292	LOX/LH2: = 99,3	Engine Propellant	
	293	LOX/LH2/kerosene: = 242,0	Engine Propellant	
	294	LOX/SINTIN: = 77,0	Engine Propellant	
	295	LOX/UDMH: = 79,5	Engine Propellant	
	296	LOX/alcohol 75% sol,: = 17,5	Engine Propellant	
	297	LOX/alcohol 92% sol,: = 22,8	Engine Propellant	
	298	LOX/kerosene: = 93,4	Engine Propellant	
	299	LOX/synthetic kerosene: = 79,4	Engine Propellant	
	300	MON1/MMH: = 7,0	Engine Propellant	
	301	MON3/AZ50: = 7,0	Engine Propellant	
	302	MON3/MMH: = 54,6	Engine Propellant	
	303	MON3/N2H4: = 1,5	Engine Propellant	
	304	N2,He: = 0,8	Engine Propellant	
	305	N2H4: = 24,8	Engine Propellant	
	306	N2H4/CIF5: = 102,0	Engine Propellant	
	307	N2H4/LF2 (Dual md): = 2,5	Engine Propellant	
	308	N2H4/MMH: = 7,5	Engine Propellant	
	309	N2O4/AZ50: = 48,3	Engine Propellant	
	310	N2O4/MMH: = 24,4	Engine Propellant	

Chamber Combustion Pressure	311	N2O4/MON: = 16,8	Engine Propellant	
	312	N2O4/N2H4: = 17,2	Engine Propellant	
	313	N2O4/UDMH: = 80,4	Engine Propellant	
	314	N2O4/UH25: = 21,0	Engine Propellant	
	315	N2O4/kerosene: = 45,0	Engine Propellant	
	316	PBAN-type: = 59,0	Engine Propellant	
Chamber Combustion Temperature	169	Tc = ((cstern* Van den Kerkhove-Faktor)^2) / R	Van den Kerkhove-Faktor	Gamma
			spezielle Gaskonstante	R
			Characteristic Velocity	cstern
Chamber Diameter	131	dchamb = ist größer Nozzle Throat Diameter	Nozzle Throat Diameter	ndt
Chamber Fuel Inlet Pressure	170	pfuelinl ist größer pc	Chamber Combustion Pressure	pc
Chamber Fuel Inlet Pressure Maximum	171	pfuelinl,max ist größer pc	Chamber Combustion Pressure	pc
Chamber Fuel Inlet Pressure Minimum	172	pfuelinl,min ist größer pc	Chamber Combustion Pressure	pc
Chamber Fuel Inlet Temperature	173	Tfuelinl ist größer pc	Chamber Combustion Pressure	pc
Chamber Inlet Pressure	174	pchamlinl ist größer pc	Chamber Combustion Pressure	pc
Chamber Inlet Pressure Maximum	175	pchaminl,max ist größer pc	Chamber Combustion Pressure	pc
Chamber Inlet Pressure Minimum	176	pchaminl,min ist größer pc	Chamber Combustion Pressure	pc
Chamber Length	725	lchamb = leng - nl	Nozzle Length	nl
			Engine Length	lengine
	726	lchamb = mcham / ((pi/4)*dchamb^2 * 0,1 * roh,al)	Dichte Aluminium	
			Kreiszahl	
			Chamber Diameter	dchamb
			Chamber Mass	mcham
	727	lchamb = Caracteristic Length*4*At / Chamber Diameter^2	Chamber Diameter	dchamb
			Characteristic Length	lstern
			Nozzle Throat Area	At
Chamber Mass	152	mcham = (pi/4)*dchamb_*lchamb* 0,1 * roh,al	Kreiszahl	
			Chamber Diameter	dchamb
			Chamber Length	lchamb
			Dichte Aluminium	
	153	mcham = mdry - mcontr-mnoz-mheatex-mpccs-mtube-mman	Controller Mass	mcontr
			Dry Mass	mdry
			Heat Exchanger Mass	mheatex
			Tube Mass	mtube
			Manifold Mass	mman
			PCCS Mass	mpccs
			Nozzle Mass	mnoz
	154	mcham = Cylindrical Chamber Mass	Cylindrical Chamber Mass	mcham,zyl
Chamber Mixture Ratio	177	O_Fcham = mOx/mFu	Propellant Fuel Mass	
			Propellant Oxidizer Mass	
	178	O_Fcham = O_F	Mixture Ratio	O_F
Chamber Oxidizer Inlet Pressure	179	poxinl ist größer pc	Chamber Combustion Pressure	pc
Chamber Oxidizer Inlet Pressure Maximum	180	poxinl,max ist größer pc	Chamber Combustion Pressure	pc
Chamber Oxidizer Inlet Pressure Minimum	181	poxinl,min ist größer pc	Adiabatenexponent	Kappa
			Chamber Combustion Pressure	pc
Chamber Wall Temperature Maximum			Nozzle Throat Pressure	pt
			Chamber Combustion Temperature	Tc
	728	Chamber Wall Temperature Maximum = Tc / ((pc/pt)^kappa-1/kappa)	Chamber Combustion Temperature	Tc
			Nozzle Throat Pressure	pt
			Adiabatenexponent	Kappa
			Chamber Combustion Pressure	pc
Characteristic Length	182	lstern = Chamber Length*Chamber Diameter_ /4*At	Nozzle Throat Area	At
			Chamber Diameter	dchamb
			Chamber Length	lchamb
Characteristic Velocity	183	cstern = ce/ cF	Thrust Coefficient	cF
			Exhaust Velocity	ce
	184	cstern = (pc*At)/ mpunkt	Mass Flow Rate	mpunkt
			Nozzle Throat Area	At
			Chamber Combustion Pressure	pc
Conceptual Design End	102	Conceptual Design End muss größer sein als Prelimenary Design Start	Prelimenary Design Start	Phase B
	103	Conceptual Design End muss kleiner sein als Conceptual Design Start	Conceptual Design Start	Phase A
Conceptual Design Start	101	Conceptual Design Start muss größer sein als Conceptual Design End	Conceptual Design End	Phase Ae
Controller Mass	155	mcontr = mdry - mcham-mnoz-mheatex-mpccs-mtube-mman	Chamber Mass	mcham
			Nozzle Mass	mnoz
			PCCS Mass	mpccs
			Manifold Mass	mman
			Tube Mass	mtube
			Heat Exchanger Mass	mheatex
			Dry Mass	mdry
Coolant Mass Flow Rate	185	mpunktcool = mpunktF (bei HS)	Fuel Mass Flow Rate	mpunktF
	186	mpunktcool = kleiner mpunktF (bei NS)	Fuel Mass Flow Rate	mpunktF
Development Start	112	Development Start muss größer sein als Ignition Test/First Hot Firing	Ignition Test/First Hot Firing	FIT
	113	Development Start muss kleiner sein als Program/Project Definition End	Program/Project Definition End	Phase Ce
Dichte der Abgase am Düsenende	74	rohe = mpunkt/ (ce*Ae)	Mass Flow Rate	mpunkt
			Nozzle Exit Area	Ae
			Exhaust Velocity	ce

Dry Mass	156	mdry = mcontr+mnoz+mman+mpccs+mtube+mcham+mheatex	Heat Exchanger Mass	mheatex
			Controller Mass	mcontr
			Manifold Mass	mman
			PCCS Mass	mpccs
			Tube Mass	mtube
			Nozzle Mass	mnoz
			Chamber Mass	mcham
Engine Diameter	132	hengine = Nozzle Exit Diameter	Nozzle Exit Diameter	nde
Engine Height	133	dengine = Engine Length	Engine Length	lengine
Engine Length	134	lengine = Nozzle Length + Chamber Length	Nozzle Length	nl
			Chamber Length	lchamb
	135	lengine = Engine Height	Engine Height	dengine
Engine Width	136	bengine = Nozzle Exit Diameter	Nozzle Exit Diameter	nde
	137	bengine = Engine Diameter	Engine Diameter	hengine
Exhaust Velocity	187	ce = Isp*9,81	Specific Impulse	Isp
	188	ce = F/ mpunkt	Mass Flow Rate	mpunkt
			Thrust	F
	189	ce = cstern *cF	Thrust Coefficient	cF
			Characteristic Velocity	cstern
	190	ce = Wurzel((2*kappa*R*Tc/((kappa-1)*M,e)*(1-(pe/pc)^((kappa-1)/kappa))	Chamber Combustion Pressure	pc
			Chamber Combustion Temperature	Tc
			spezielle Gaskonstante	R
			Nozzle Exit Pressure	pe
			Adiabatenexponent	Kappa
			Mittlere Molare Masse des Abgases	
Final Documentation/Report	130	Final Documentation/Report muss kleiner sein als Last Flight	Last Flight	LF
First Flight Test	122	First Flight Test muss größer sein als Production Start	Production Start	Phase E
	123	First Flight Test muss kleiner sein als Qualification Test End	Qualification Test End	QTE
First Full Thrust & Full Burn Time Test	118	First Full Thrust & Full Burn Time Test muss größer sein als Qualification Test End	Qualification Test End	QTE
	119	First Full Thrust & Full Burn Time Test muss kleiner sein als First Full Thrust Test	First Full Thrust Test	FTT
First Full Thrust Test	116	First Full Thrust Test muss größer sein als First Full Thrust & Full Burn Time Test	First Full Thrust & Full Burn Time Test	FT&FBT
	117	First Full Thrust Test muss kleiner sein als Ignition Test/First Hot Firing	Ignition Test/First Hot Firing	FIT
Fuel Mass Flow Rate	192	mpunktF = mpunktox * mixture ratio	Mixture Ratio	O_F
			Oxidizer Mass Flow Rate	mpunktox
	193	mpunktF = mpunkt-mpunktox	Oxidizer Mass Flow Rate	mpunktox
			Mass Flow Rate	mpunkt
	194	mpunktF = mfuel / BurnTime	Propellant Fuel Mass	
			Pulse Duration	tIbit
			Burn Time	tc
Heat Exchanger Mass	157	mheatex = mdry - mcontr-mnoz-mman-mpccs-mtube-mcham	Nozzle Mass	mnoz
			Chamber Mass	mcham
			Controller Mass	mcontr
			Dry Mass	mdry
			Tube Mass	mtube
			Manifold Mass	mman
			PCCS Mass	mpccs
Ignition Test/First Hot Firing	114	Ignition Test/First Hot Firing muss größer sein als First Full Thrust Test	First Full Thrust Test	FTT
	115	Ignition Test/First Hot Firing muss kleiner sein als Development Start	Development Start	Phase D
Impulse Bit Minimum	729	Impulse Bit Minimum <= Pulse Duration	Pulse Duration	tIbit
Last Flight	128	Last Flight muss größer sein als Final Documentation/Report	Final Documentation/Report	FR
	129	Last Flight muss kleiner sein als Production End	Production End	Phase Ee
Manifold Mass	158	mman = mdry - mcontr-mnoz-mheatex-mpccs-mtube-mcham	PCCS Mass	mpccs
			Tube Mass	mtube
			Heat Exchanger Mass	mheatex
			Controller Mass	mcontr
			Dry Mass	mdry
			Chamber Mass	mcham
			Nozzle Mass	mnoz
Mass Flow Rate	195	mpunkt = F/ ce	Exhaust Velocity	ce
			Thrust	F
	196	mpunkt = pc*At/cstern	Nozzle Throat Area	At
			Chamber Combustion Pressure	pc
			Characteristic Velocity	cstern
	197	mpunkt = mpunkt_At *At	Nozzle Throat Area	At
			Specific Mass Flow Rate	mpunkt_At
	198	mpunkt = rpunkt*Ap*rohp	Treibstoffdichte	rohp
			Abbrandgeschwindigkeit	rpunkt
			Abbrandoberfläche	Ap
	199	mpunkt = m8/Za	Aktionszeit	Za
			Usable Propellant Mass	
	200	mpunkt = rohe*Ae*ce	Nozzle Exit Area	Ae
			Dichte der Abgase am Düsenende	rohe
			Exhaust Velocity	ce
	201	mpunkt = 2*Psp/ce	Exhaust Velocity	ce
			Specific Power	Psp

Mass Flow Rate	202	mpunkt = mpunktF+mpunktox	Fuel Mass Flow Rate Oxidizer Mass Flow Rate	mpunktF mpunktox
	203	mpunkt = m8/t(c*NOE)	Number of engines Burn Time Usable Propellant Mass	 tc
Mixture Ratio	204	O_F = mox/mfuel	Propellant Fuel Mass Propellant Oxidizer Mass	
	205	O_F = mpunktOx/mpunktF	Fuel Mass Flow Rate Oxidizer Mass Flow Rate	mpunktF mpunktox
	206	O_F = (O_Fmax+O_Fmin)/2	Mixture Ratio Maximum Mixture Ratio Minimum	O_Fmax O_Fmin
	207	O_F = O_Fcham	Chamber Mixture Ratio	O_Fcham
	243	F2/NH3: = 2,7	Engine Propellant	
	244	GOX/synthetic kerosene: = 3,5	Engine Propellant	
	245	HNO3/N2H4: = 1,5	Engine Propellant	
	246	HNO3/UDMH: = 2,6	Engine Propellant	
	247	HNO3/amine based: = 3,1	Engine Propellant	
	248	HNO3/kerosene: = 4,0	Engine Propellant	
	249	HTPB-type: = 2,3	Engine Propellant	
	250	IRFNA/0,65Aniline+0,35Furfuryl Alcohol: = 2,6	Engine Propellant	
	251	IRFNA/UDMH: = 2,8	Engine Propellant	
	252	LF2/LNH3: = 2,7	Engine Propellant	
	253	LOX/CH4: = 3,3	Engine Propellant	
	254	LOX/GH2: = 5,0	Engine Propellant	
	255	LOX/LH2: = 5,5	Engine Propellant	
	256	LOX/LH2/kerosene: = 4,0	Engine Propellant	
	257	LOX/UDMH: = 1,7	Engine Propellant	
	258	LOX/kerosene: = 2,5	Engine Propellant	
	259	MON1/MMH: = 1,7	Engine Propellant	
	260	MON3/AZ50: = 1,6	Engine Propellant	
	261	MON3/MMH: = 2,0	Engine Propellant	
	262	MON3/N2H4: = 0,8	Engine Propellant	
	263	N2H4/CIF5: = 2,7	Engine Propellant	
	264	N2H4/MMH: = 1,4	Engine Propellant	
	265	N2O4/AZ50: = 1,9	Engine Propellant	
	266	N2O4/MMH: = 1,7	Engine Propellant	
	267	N2O4/MON: = 1,6	Engine Propellant	
	268	N2O4/N2H4: = 1,5	Engine Propellant	
	269	N2O4/UDMH: = 2,2	Engine Propellant	
	270	N2O4/UH25: = 1,8	Engine Propellant	
	271	N2O4/kerosene: = 4,0	Engine Propellant	
Mixture Ratio Maximum	209	O_Fmax = O_F	Mixture Ratio	O_F
Mixture Ratio Minimum	210	O_Fmin = O_F	Mixture Ratio	O_F
Nozzle Area Ratio	211	Ae_At = Ae/At	Nozzle Exit Area	Ae
	241	Ae_At = 10 (bei Boosterbetrieb)		
	242	Ae_At = 20 (bei Vakuumbetrieb)		
Nozzle Area Ratio End	212	Ae_At,e = oder größer Ae_At	Nozzle Area Ratio	Ae_At
Nozzle Exit Area	138	Ae = (Fsl - mpunkt * ce)/ (pe-pa)	Mass Flow Rate Nozzle Exit Pressure Thrust Sea Level Exhaust Velocity Sea Level Pressure	mpunkt pe Fsl ce
	139	Ae = (Fvac - mpunkt * ce) / pe	Exhaust Velocity Thrust Vacuum Mass Flow Rate Nozzle Exit Pressure	ce Fvac mpunkt pe
	140	Ae = Ae_At*At	Nozzle Area Ratio Nozzle Throat Area	Ae_At At
	141	Ae = mpunkt/ (pc* Wurzel ((2*Kappa*M,e / (Kappa -1) * R *Tc)) * (((pc/pe)*exp(-2/kappa)) - (pc/pe)*exp(-1-Kappa/Kappa))	Nozzle Exit Pressure Adiabatenexponent spezielle Gaskonstante Chamber Combustion Pressure Chamber Combustion Temperature Mass Flow Rate Mittlere Molare Masse des Abgases	pe Kappa R pc Tc mpunkt
	142	Ae ist größer Nozzle Throat Area	Nozzle Throat Area	At
	143	Ae = pi/4*nde^2	Nozzle Exit Diameter	nde
Nozzle Exit Diameter	144	nde = wurzel((Ae*4)/pi)	Nozzle Exit Area	Ae
	145	nde = nl*tan (alpha) + ndt	halber Öffnungswinkel Nozzle Length Nozzle Throat Diameter	alpha nl ndt
Nozzle Exit Pressure	213	pe = pc/ pc_pe	Chamber Combustion Pressure Expansion Ratio	pc
	730	Pe = 0,3 bar bei Erststufentriebwerken	Stage Number	
Nozzle Expansion Ratio	214	pc_pe = pc/ pe	Chamber Combustion Pressure Nozzle Exit Pressure	pc pe

Nozzle Length	146	nl = (nde - ndt) / tan (alpha)	Nozzle Throat Diameter	ndt
			halber Öffnungswinkel	alpha
			Nozzle Exit Diameter	nde
Nozzle Mass	62	tc = 0,03*Wurzel(m8*nl/Za)	Treibstoffmasse	
			Nozzle Length	nl
			Aktionszeit	Za
	159	mnoz = mdry - mcontr-mman-mpccs-mtube-mcham-mheatex	Chamber Mass	mcham
			Tube Mass	mtube
			Manifold Mass	mman
			PCCS Mass	mpccs
			Controller Mass	mcontr
			Dry Mass	mdry
			Heat Exchanger Mass	mheatex
Nozzle Throat Area	147	At = cstern*mpunkt/pc	Mass Flow Rate	mpunkt
			Caracteristic Velocity	
			Chamber Combustion Pressure	pc
	148	At = Ae/ Ae_At	Nozzle Area Ratio	Ae_At
			Nozzle Exit Area	Ae
	149	At = pi/4*ndt^2	Nozzle Throat Diameter	ndt
Nozzle Throat Diameter	150	ndt = wurzel((At*4)/pi)	Nozzle Throat Area	At
	151	ndt = nl*tan (alpha) - nde	Nozzle Length	nl
			halber Öffnungswinkel	alpha
			Nozzle Exit Diameter	nde
Nozzle Throat Pressure	42	pt = pc / (2/Kappa+1) * exp(Kappa/1-Kappa)	Chamber Combustion Pressure	pc
			Adiabatenexponent	Kappa
Nozzle Throat Velocity	43	vt = Wurzel((2*kappa*R*Tc)/ ((kappa+1)*Mmol,e))	Adiabatenexponent	Kappa
			Chamber Combustion Temperature	Tc
			universelle Gaskonstante(8314000)	
			Mittlere Molare Masse des Abgases	
Oxidizer Mass Flow Rate	216	mpunktox = mpunktox / mixture ratio	Mixture Ratio	O_F
			Oxidizer Mass Flow Rate	mpunktox
	217	mpunktox = mpunkt-mpunktF	Impulse Bit Minimum	tIbit,min
			Mass Flow Rate	mpunkt
			Fuel Mass Flow Rate	mpunktF
PCCS Mass	160	mpccs = mdry - mcontr-mnoz-mman-mheatex-mtube-mcham	Heat Exchanger Mass	mheatex
			Controller Mass	mcontr
			Dry Mass	mdry
			Manifold Mass	mman
			Tube Mass	mtube
			Chamber Mass	mcham
			Nozzle Mass	mnoz
Prelimenary Design End	106	Prelimenary Design End muss größer sein als Program/Project Definition Start	Program/Project Definition Start	Phase C
	107	Prelimenary Design End muss kleiner sein als Prelimenary Design Start	Prelimenary Design Start	Phase B
Prelimenary Design Start	104	Prelimenary Design Start muss größer sein als Prelimenary Design End	Prelimenary Design End	Phase Be
	105	Prelimenary Design Start muss kleiner sein als Conceptual Design End	Conceptual Design End	Phase Ae
Production End	126	Production End muss größer sein als Last Flight	Last Flight	LF
	127	Production End muss kleiner sein als Production Start	Production Start	Phase E
Production Start	124	Production Start muss größer sein als Production End	Production End	Phase Ee
	125	Production Start muss kleiner sein als First Flight Test	First Flight Test	FFT
Program/Project Definition End	110	Program/Project Definition End muss größer sein als Development Start	Development Start	Phase D
	111	Program/Project Definition End muss kleiner sein als Program/Project Definition Start	Program/Project Definition Start	Phase C
Program/Project Definition Start	108	Program/Project Definition Start muss größer sein als Program/Project Definition End	Program/Project Definition End	Phase Ce
	109	Program/Project Definition Start muss kleiner sein als Prelimenary Design End	Prelimenary Design End	Phase Be
Qualification Test End	120	Qualification Test End muss größer sein als First Flight Test	First Flight Test	FFT
	121	Qualification Test End muss kleiner sein als First Full Thrust & Full Burn Time Test	First Full Thrust & Full Burn Time Test	FT&FBT
Specific Impulse	219	Isp = ce/9,81	Exhaust Velocity	ce
Specific Impulse Sea Level	220	Isp,SL = Fsl / mpunkt	Mass Flow Rate	mpunkt
			Thrust Sea Level	Fsl
Specific Impulse Vacuum	221	Isp,Vac = Fvac / mpunkt	Thrust Vacuum	Fvac
			Mass Flow Rate	mpunkt
Specific Mass Flow Rate	37	mpunkt_At = mpunkt / At	Mass Flow Rate	mpunkt
			Nozzle Throat Area	At
	38	mpunkt_At = pc / cstern	Chamber Combustion Pressure	pc
			Characteristic Velocity	cstern
			Chamber Combustion Pressure	pc
	39	mpunkt_At = pc * Van den Kerkove-Faktor/ Wurzel (R*Tc/M,e)	Chamber Combustion Pressure	pc
			Chamber Combustion Temperature	Tc
			Van den Kerkhove-Faktor	Gamma
			universelle Gaskonstante(8314000)	
			Mittlere Molare Masse des Abgases	
Specific Power	222	Psp = (mpunkt*ce^2) / 2	Mass Flow Rate	mpunkt
			Exhaust Velocity	ce
Thrust	223	F = mpunkt * ce	Exhaust Velocity	ce
			Mass Flow Rate	mpunkt
Thrust Coefficient	317	cF = ce/ cstern	Exhaust Velocity	ce

			Caracteristic Velocity	
	318	cF = (pc*At)/ mpunkt	Chamber Combustion Pressure	pc
			Nozzle Throat Area	At
			Mass Flow Rate	mpunkt
Thrust Coefficient Sea Level	50	cFsl = cFvac - (Ae*psl/(At*pc))	Nozzle Exit Area	Ae
			Pressure Sea Level	
			Thrust Coefficient Vacuum	cFvac
			Nozzle Throat Area	At
			Chamber Combustion Pressure	pc
Thrust Coefficient Vacuum	49	cFvac = Wurzel (2*Kappa*exp2/Kappa-1) * (2/Kappa+1)*exp(Kappa+1/Kappa-1)	Adiabatenexponent	Kappa
Thrust Maximum	224	Fmax = F	Thrust	F
Thrust Minimum	225	Fmin = F	Thrust	F
Thrust Sea Level	226	Fsl = mpunkt * ce + Ae * (pe-pa)	Mass Flow Rate	mpunkt
			Nozzle Exit Area	Ae
			Exhaust Velocity	ce
			Sea Level Pressure	
			Nozzle Exit Pressure	pe
	227	Fsl = 0,9 * Fvac	Thrust Vacuum	Fvac
Thrust Vacuum	228	Fvac = mpunkt * ce + Ae * pe	Mass Flow Rate	mpunkt
			Nozzle Exit Area	Ae
			Exhaust Velocity	ce
			Nozzle Exit Pressure	pe
Thrust Vacuum Max	229	Fmax,Vac = Fvac	Thrust Vacuum	Fvac
Thrust Vacuum Min	230	Fmin,Vac = Fvac	Thrust Vacuum	Fvac
Total Mass	161	mtot = m4 / NOE	Propulsion System Mass	
			Number Of Engines	Noe
Treibstoffdichte	54	rohp = mpunkt / (Ap*rpunkt)	Abbrandgeschwindigkeit	rpunkt
			Abbrandoberfläche	Ap
			Mass Flow Rate	mpunkt
Tube Mass	162	mtube = mdry - mcontr-mnoz-mman-mheatex-mpccs-mcham	Manifold Mass	mman
			PCCS Mass	mpccs
			Heat Exchanger Mass	mheatex
			Controller Mass	mcontr
			Dry Mass	mdry
			Nozzle Mass	mnoz
			Chamber Mass	mcham
Van den Kerkhove-Faktor	27	Van den Kerkhove-Faktor = Wurzel (Kappa * (2/ Kappa+1)*exp ((Kappa+1) /2*(Kappa-1))	Adiabatenexponent	Kappa
	168	pc = (2/Kappa+1) * exp(Kappa/1-Kappa)	Adiabatenexponent	Kappa
Webb-Thickness	56	b = rpunkt*tc	Abbrandgeschwindigkeit	rpunkt
			Burn Time	tc
Zündverzugszeit	231	deltaZ = 0,02		
	232	deltaZ = 0,2		
halber Öffnungswinkel	47	alpha = arctan ((nde-ndt) /nl)	Nozzle Length	nl
			Nozzle Throat Diameter	ndt
			Nozzle Exit Diamater	
spezielle Gaskonstante	71	R = Rr/ M,c	universelle Gaskonstante	
Äussere Klemmung	51	Ka = Ap/At	Nozzle Throat Area	At
			Abbrandoberfläche	Ap
	236	Ka = 225		
	237	Ka = 350		

Sonstige

Regel_ID	Regelname
0	Einheiten in Ordnung, keine Einheitenumrechnung erforderlich
1	Wert in Entity-Tabelle vorhanden
9007	Wenn 1 bis n default = n Werte und kein default = y Wert dann Mittelwert aus den default = y Werten bilden
9008	Wenn genau 1 default = y Wert dann diesen Wert nehmen
9009	Wenn 2 bis n default = y Werte dann Mittelwert aus den default = y Werten bilden

10.4 Programmlistings

Start Oracle

```
'<LOAD>'(D):-
   ensure_loaded('Oracle Boot.PC'),
   db_close,
   db_open(oracle(aim,password)),
   abolish('<LOAD>'/1),
   consult(engine_tables), consult(stage_tables), consult(launcher_tables),
   consult(engine_rules1), consult(engine_rules2),
   consult(stage_rules1), consult(stage_rules2),
   consult(launcher_rules1), consult(launcher_rules2),
   wkill('Start ORACLE').
```

Stop Oracle

```
'<LOAD>'(D):-
   ensure_loaded('Oracle Boot.PC'),
   db_close,
   abolish('<LOAD>'/1),
   wkill('Stop ORACLE').
```

Start Prolog

```
%---------------------------
% HyperCard Übergabe
%---------------------------
% ObjectClass=[launcher,stage,engine,…]
% Property=['Thrust','Specific Impulse','Mass Flow Rate',…]
% Identifier=[Engine_ID,Stage_ID,Launcher_ID,…]
hyp_asks_prolog(ObjectClass,Property,Identifier):-
   wkill(hyp_to_prol),
   retractall(used(A1,A2)), retractall(wanted(C1)),
   asserta(wanted(Property)),
   db_reset_cursors, db_flag(show_db_error,Old,off),
   forall(ObjectClass(Property,Identifier,Answer,Unit,Regel,-1),ausgabe(Answer,Unit,Regel)),!.
hyp_asks_prolog(ObjectClass,Property,Identifier,Anzahl):-
   wkill(hyp_to_prol),
   retractall(used(A1,A2)), retractall(wanted(C1)),
   asserta(wanted(Property)),
   db_reset_cursors, db_flag(show_db_error,Old,off),
   ObjectClass(Property,Identifier,Answer,Unit,Regel,-1),
   not Answer=[],
   % Normaler Output
   write('['), write(Answer), write(', '), write(Unit), write(', '), write(Regel), writenl(']'),
   % Output in Datei
   fcreate('prol_to_hyp',0), fopen('prol_to_hyp',1), output('prol_to_hyp'),
   write('['), write(Answer), write(', '), write(Unit), write(', '), write(Regel), writenl(']'),
   fclose('prol_to_hyp'),
   launch_app('HyperCard',switch,continue,PSN),!.
ausgabe(A,U,R):-
   % Normaler Output
   write('['), write(A), write(', '), write(U), write(', '), write(R), writenl(']'),
   % Output in Datei
   fcreate('prol_to_hyp',0), fopen('prol_to_hyp',1), output('prol_to_hyp'),
   write('['), write(A), write(', '), write(U), write(', '), write(R), writenl(']'),
   fclose('prol_to_hyp').
%---------------------------
% NO SOLUTION Ausnahme
%---------------------------
hyp_asks_prolog(ObjectClass,Property,Identifier):-
   writenl('no solution'), % Normaler Output
   fcreate('prol_to_hyp',0), % Output in Datei
   fopen('prol_to_hyp',1),
   output('prol_to_hyp'), writenl('no solution'),
   fclose('prol_to_hyp'),
   launch_app('HyperCard',switch,continue,PSN).
hyp_asks_prolog(ObjectClass,Property,Identifier,_):-
   writenl('no solution'), % Normaler Output
   fcreate('prol_to_hyp',0), % Output in Datei
   fopen('prol_to_hyp',1),
   output('prol_to_hyp'), writenl('no solution'),
   fclose('prol_to_hyp'),
   launch_app('HyperCard',switch,continue,PSN).
```

```
%---------------------------
% Einheiten Umrechnung
%---------------------------
%Druck
check_unit(_,Value,'N/m^2',Value,'N/m^2',0).
check_unit(_,Value,'Pa',Value,'N/m^2',108).
check_unit(_,Value,'bar',NewValue,'N/m^2',101):-
   NewValue is Value * 100000.
%Kraft
check_unit(_,Value,'N',Value,'N',0).
check_unit(_,Value,'kN',NewValue,'N',102):-
   NewValue is Value * 1000.
%Masse
check_unit(_,Value,'kg',Value,'kg',0).
check_unit(_,Value,'g',NewValue,'kg',103):-
   NewValue is Value / 1000.
check_unit(_,Value,'t',NewValue,'kg',104):-
   NewValue is Value * 1000.
%Länge
check_unit(_,Value,'m',Value,'m',0).
check_unit(_,Value,'km',NewValue,'m',105):-
   NewValue is Value * 1000.
check_unit(_,Value,'cm',NewValue,'m',106):-
   NewValue is Value / 100.
check_unit(_,Value,'mm',NewValue,'m',107):-
   NewValue is Value / 1000.
%Fläche
check_unit(_,Value,'m^2',Value,'m^2',0).
%Massenfluss
check_unit(_,Value,'kg/s',Value,'kg/s',0).
%Volumen
check_unit(_,Value,'m^3',Value,'m^3',0).
%Zeit
% Ausnahmen für spezifischen Impuls in s
check_unit('Specific Impulse',Value,'s',NewValue,'m/s',1000):-
   NewValue is Value * 9.81,!.
check_unit('Specific Impulse Vacuum',Value,'s',NewValue,'m/s',1000):-
   NewValue is Value * 9.81,!.
check_unit('Specific Impulse Sea Level',Value,'s',NewValue,'m/s',1000):-
   NewValue is Value * 9.81,!.
% normale Zeitumrechnungen
check_unit(_,Value,'s',Value,'s',0).
check_unit(_,Value,'min',NewValue,'s',108):-
   NewValue is Value / 60.
check_unit(_,Value,'h',NewValue,'s',108):-
   NewValue is Value / 3600.
%Geschwindigkeit
check_unit(_,Value,'m/s',Value,'m/s',0).
check_unit(_,Value,'km/s',NewValue,'m/s',108):-
   NewValue is Value * 1000.
```

```
%-----------------------------
% sonstige Regeln
%-----------------------------
not_used(X,Y):-
   not used(X,Y),
   not used(Y,X).
not_wanted(X):-
   wanted(Y),
   not member(Y,X).
liste_in_werte([X],[A],Class,Attribut,Unit,[Rule]):-
   R is X+10000,
   Class(Attribut,X,A,Unit,Rule,R).
liste_in_werte([X|Y],[A|Z],Class,Attribut,Unit,[Rule|Z1]):-
   R is X+10000,
   Class(Attribut,X,A,Unit,Rule,R),
   liste_in_werte(Y,Z,Class,Attribut,Unit,Z1).
%---------------
% Minimum
min(L,Min):-
   sort(L,[Min|Rest]).
%---------------
% Maximum
max(L,Max):-
   sort(L,L_sort),
   reverse(L_sort,[Max|Rest]).
%---------------
% Summe
summe([El1|Rest],Summe):-
   add(El1,Rest,Summe).
add(Summe,[],Summe).
add(Sum_alt,[Next|Rest],Summe):-
   Sum_neu is Sum_alt+Next,
   add(Sum_neu,Rest,Summe).
%---------------
% Produkt
produkt([El1|Rest],Prod):-
   mult(El1,Rest,Prod).
mult(Prod,[],Prod).
mult(Prod_alt,[Next|Rest],Prod):-
   Prod_neu is Prod_alt*Next,
   mult(Prod_neu,Rest,Prod).
%---------------
% X Vorhanden ?
vorhanden(El,Liste,yes):-
    member(El,Liste).
vorhanden(El,Liste,no):-
    not member(El,Liste).
%---------------
% Alle = X ?
alle_gleich(Liste,yes):-
    setof(El,member(El,Liste),[L]).
alle_gleich(Liste,no):-
    setof(El,member(El,Liste),[El1,El2|Rest]).
%---------------------------
% Mittelwertbildung
%---------------------------
proceed(Attribut,Liste,W,Unit,Regel):-
   proc(Attribut,Liste,[[V,U]|Rneu],Unit,Regel),
   length([[V,U]|Rneu],Len),
   sum(0,[[V,U]|Rneu],Sum),
   W is Sum/Len.
proc(Attribut,[[V1,U1]|R],[[V,U]|Rneu],U,[Reg|Regel]):-
   check_unit(Attribut,V1,U1,V,U,Reg),
   proc(Attribut,R,Rneu,U,Regel).
proc(Attribut,[],[],U,[]).
-------------------------------
% Berechnung der Summe einer Liste
sum(S0,[[V|U]|R],S):-
   S0_neu is S0+V,
   sum(S0_neu,R,S).
sum(S,[],S). % Abbruchbedingung für Summenberechnung
%---------------------------
% Konstanten
%---------------------------
pi(3.14592654).
e(2.71818281828).
univ_Gaskonst(8314000). % [J/(kg * mol)]
pa(10130). % [N/m^2]
```